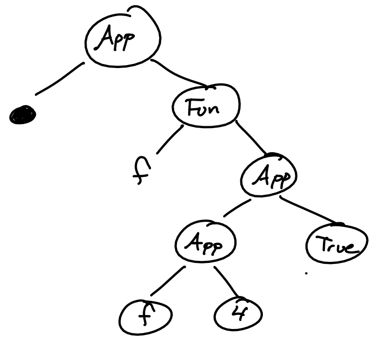
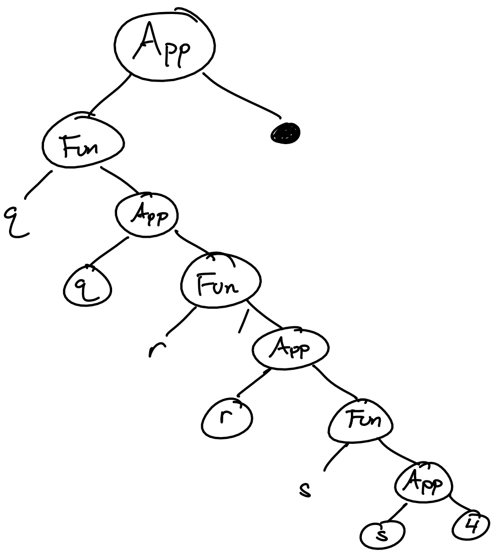
* Classes
* Example of application w/ func arg
* ASTs for practice exam context questions
* Subtyping, esp. functions
* Mutual recursion

Practice Exam Q1 ASTs

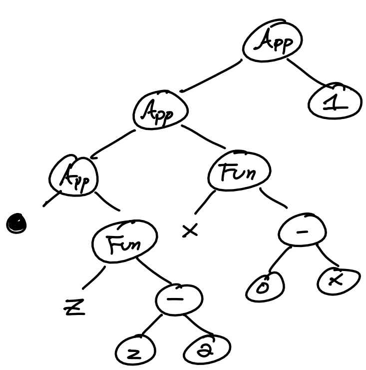
a.



b.



c.

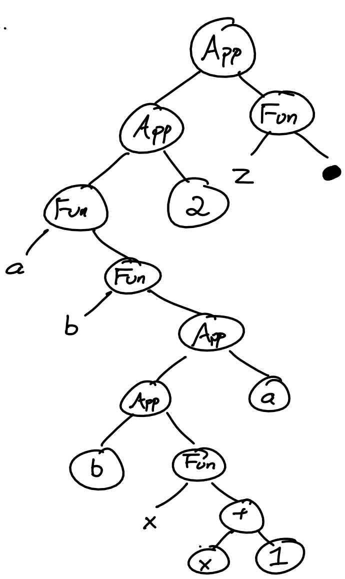


$(1+2)+3$

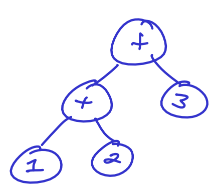
$(f\ g)\ h$

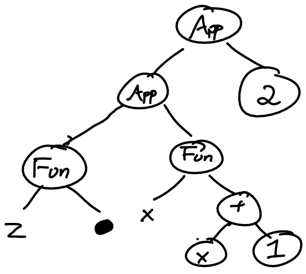
$f \circ x \circ y$

d.

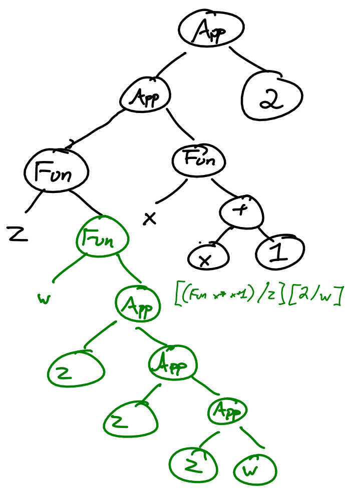


$(1+2)+3$





(Fun z → •) (Fun x → x+1) 2



z(z(z w))

(Fun w → z (z (z w)))

$$\big((\text{Function } f \to \text{Function } x \to f\,(f\,x))\ (\text{Function } n \to n+1)\big)\,3$$

$$\dfrac{\dfrac{}{(F\ n\to n+1) \Rightarrow (F\ n \to n+1)} \qquad \dfrac{V\dfrac{}{(F\ n\to n+1)\Rightarrow \cdots} \quad V\dfrac{}{3\Rightarrow 3} \quad A\dfrac{\overline{3\Rightarrow 3}\quad \overline{1\Rightarrow 1}}{3+1\Rightarrow 4}}{\underbrace{(F\ n\to n+1)}_{e_1}\ \underbrace{3}_{e_2} \Rightarrow 4} \qquad A\dfrac{V\dfrac{}{4\Rightarrow 4}\quad V\dfrac{}{1\Rightarrow 1}}{4+1 \Rightarrow 5}}{\underbrace{(F\ n\to n+1)}_{e_1}\ \underbrace{((F\ n\to n+1)\ 3)}_{e_2} \Rightarrow 5}$$

$$\dfrac{\dfrac{V\dfrac{}{(F\ f\to\cdots)\Rightarrow(F\ f\to\cdots)} \quad V\dfrac{}{(F\ n\to n+1)\Rightarrow(F\ n\to n+1)} \quad V\dfrac{}{(F\ x\to (F\ n\to n+1)\,((F\ n\to n+1)\ x))\Rightarrow\cdots}}{\underbrace{(F\ f\to F\ x\to f\,(f\,x))}_{e_1}\ \underbrace{(F\ n\to n+1)}_{e_2} \Rightarrow (F\ x\to (F\ n\to n+1)\,((F\ n\to n+1)\ x))} \qquad V\dfrac{}{3\Rightarrow 3}}{\underbrace{((F\ f\to F\ x\to f\,(f\,x))\ (F\ n\to n+1))}_{e_1}\ \underbrace{3}_{e_2} \Rightarrow 5}$$

Classes encoded in FbSR

```
class Counter:
  def __init__(self):
      self.n = 0
  def next(self):
      m = self.n
      self.n += 1
      return m
  z = 4

c = Counter()
c.next()
```

Make An Object

```
Let counter = { __init__ =
                  Function this ->
                     { n = Ref 0;
                       next = Function this ->
                                 Let m = !this.n  In
                                 Let junk = (this.n := !this.n + 1) In
                                 m
                     };
                z = Ref 4
              }
In
Let c = counter.__init__ counter In
c.next c
```

Private Fields

```
Let counter = { __init__ =
          Function this ->
             Let private = { n = Ref 0 } In

             {
               next = Function this ->
                         Let m = !private.n  In
                         Let junk = (private.n := !private.n + 1) In
                         m
             };
     z = Ref 4
   }
In
Let c = counter.__init__ counter In
```

```
class Counter {
  private int n;
  public Counter() {
     this.n = 0;
  }
  public int next() {
     ...
  }
}
```

```
Let fresh =
   e_1 [ Let ctr = Ref 0 In
         Function junk ->
            ctr := !ctr + 1 ]
In
```

Cells are values

No concrete syntax directly creates a particular cell

$$\frac{\frac{}{\langle \emptyset, 0\rangle \Rightarrow \langle \emptyset, 0\rangle}}{\langle \emptyset, \text{Ref } 0\rangle \Rightarrow \langle \{\#1 \mapsto 0\}, \#1\rangle} \qquad \frac{\vee}{\langle \{\#1 \mapsto 0\}, \text{Fun junk} \to \#1 := !\#1 + 1\rangle \Rightarrow \cdots}$$

$$\overline{\langle \emptyset, \text{Let ctr} = \text{Ref } 0 \text{ In } \underline{\text{Fun junk} \to \text{ctr} := !\text{ctr} + 1}\rangle \Rightarrow \langle \{\#1 \mapsto 0\}\ \text{Fun junk} \to \#1 := !\#1 + 1\rangle \Rightarrow}$$

Poodle <: Dog

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}\ \text{SubFun}$$

Dog → ... <: Poodle → ...

$\{a : Int; b : Int\} <: \{a : Int\}$

Poodle <: Dog

"Dogsitter" : Dog → ·

Poodle → ·

STFbR

$\Gamma \vdash e : \tau$

$\tau <: \tau$

$$\boxed{\frac{\Gamma \vdash e : \tau_1 \qquad \tau_1 <: \tau_2}{\Gamma \vdash e : \tau_2}}$$

Dog* d = new Poodle();

$- \longrightarrow +$

$(\overset{-}{\cdot} \to \overset{+}{\cdot}) \to (\overset{-}{\cdot} \to \overset{+}{\cdot})$

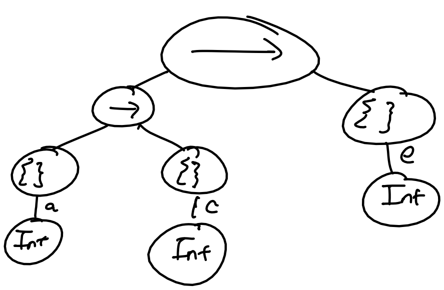
$+$ co, $-$ contra, $-$ contra, $+$ co

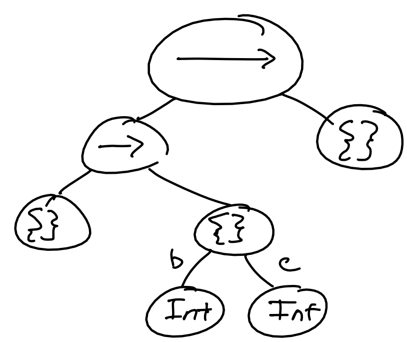
$\tau_4' <: \tau_4,$
$\tau_3 <: \tau_3'$
$\tau_2 <: \tau_2'$
$\tau_1' <: \tau_1$

General deep subtyping for functions:

$$\frac{\text{SubFun}\ \dfrac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'} \qquad \text{SubFun}\ \dfrac{\tau_3 <: \tau_3' \qquad \tau_4' <: \tau_4}{\tau_3' \to \tau_4' <: \tau_3 \to \tau_4}}{(\tau_1' \to \tau_2') \to (\tau_3' \to \tau_4') <: (\tau_1 \to \tau_2) \to (\tau_3 \to \tau_4)}\ \text{SubFun}$$

$(\{a : Int\} \to \{c : Int\}) \to \{e : Int\}$ $\qquad$ $(\{\} \to \{b : Int; c : Int\}) \to \{\}$





$$\frac{\text{SF}\ \dfrac{\text{SR}\ \dfrac{}{\{a : Int\} <: \{\}} \qquad \text{SR}\ \dfrac{\text{Refl}\ \dfrac{}{Int <: Int}}{\{b : Int; c : Int\} <: \{c : Int\}}}{\{\} \to \{b : Int; c : Int\} <: \{a : Int\} \to \{c : Int\}} \qquad \text{SR}\ \dfrac{}{\{e : Int\} <: \{\}}}{(\{a : Int\} \to \{c : Int\}) \to \{e : Int\} <: (\{\} \to \{b : Int; c : Int\}) \to \{\}}\ \text{SF}$$

$$\text{SR}\ \frac{\tau_1 <: \tau_1' \qquad \cdots \qquad \tau_n <: \tau_n'}{\{l_1 : \tau_1, \ldots, l_n : \tau_n, \ldots l_m : \tau_m\} <: \{l_1 : \tau_1', \ldots, l_n : \tau_n'\}} \qquad \text{Refl}\ \frac{}{\tau <: \tau}$$

```
Let countdown' = Function self -> Function n ->
                        ... self self (n-1) ...

                foo'              bar'            arg
Let foo' = Function self -> Function other -> Function x ->
           If x = 0 Then 0 Else other other self (x-1)
                bar'              foo'            arg
In
Let bar' = Function self -> Function other -> Function x ->
           If x = 0 Then 0 Else
           If x = 1 Then 1 Else
           other other self (x-2)

In
foo' foo' bar' 5
```

```
(Function f1 -> Function f2 ->
  Let callf1' = Function self -> Function other -> Function n ->
        f1 (self self other) (other other self) n
  In
  Let callf2' = Function self -> Function other -> Function n ->
        f2 (self self other) (other other self) n
  In
  Let callf1 = callf1' callf1' callf2' In
  Let callf2 = callf2' callf2' callf1' In
  f1 callf1 callf2
)
```

(Function q -> q (Function r -> r (Function s -> s 4)))•·

•(Function r → r (Function s → s 4))

(Function rfn → rfn (Function sfn → sfn (Function n → n))) (Function r → r (Function s → s 4))

Operational Equivalence: $e_1 \cong e_2$ iff $\forall C.\ C[e_1] \Rightarrow v_1$ iff $C[e_2] \Rightarrow v_2$

$$C' = \text{If } C[\bullet] = v_1 \text{ Then } 0 \text{ Else } \Omega$$

To prove $e_1 \not\cong e_2$, find one counterexample.
To prove $e_1 \cong e_2$, full formal proof!

FbM

$e ::= \cdots \mid \text{Some } e \mid \text{Default } e / x \to e / e$
$v ::= \cdots \mid \text{Some } v \mid \text{None}$

Some a　　　Some(1+2)
Some b　　　Sone (f x)
None

1+2+3

a=1
b=2
c=a+b
d=3
e=c+d

$$\frac{e_1 \Rightarrow \text{None} \quad e_3 \Rightarrow v}{\text{Default } e_1 / x \to e_2 / e_3 \Rightarrow v}$$

$$\frac{e_1 \Rightarrow \text{Some } v' \quad e_2[v'/x] \Rightarrow v}{\text{Default } e_1 / x \to e_2 / e_3 \Rightarrow v}$$

(Function a → Default a / n → n / 0) (Some 5)

Match e With
| Some x → e
| None → e