2-1 J_1 $e ::= v \mid (e\ e\ \ldots) \mid (\text{if } e\ e\ e)$

$v ::= b$

$b ::= \text{num} \mid \text{bool} \mid \text{prim}$

$\text{prim} ::= +, -, *, /, \leq, <, =, >, \geq, \ldots$

big step semantics interp : $e \to v$

interp $(/\ 1\ 0) = \bot$ $\delta(/, 1, 0) = \bot$

math isn't like this!

equational theory $a + b = b + a$

$1 + 1 = 2$

$2+3 = 2+2+1 = 4+1 = 5$

2-2) small-step evaluation

big step

$$1 + (2 \times 3) + (6 \times 10)$$
$$1 + 6 + (6 \times 10)$$
$$1 + 6 + 60$$
$$7 + 60$$
$$67$$

small step

done when get v or there's no steps left

```
interp e =
  let e' = step e
  if e == e' then
      return e
  o.w. interp e'
```

2-3/ step : e $\to$ e

evaluation rules

step (if v et ef) = et where $v \neq$ false

step (if false et ef) = ef

step (p v ...) = δ(p, v ...)

step v = v

step (if e $\neq$ v et ef)

= (if (step e) et ef)

step (v ... e $\neq$ v earg ...) =

(v ... (step e) earg ...)

congruence rules or structural rules

2-4/ A context is a "program with a hole".

```
C ::= hole — ■
   | (if C e e)          if1C
   | (if e C e)          if2C
   | (if e e C)          if3C
   | (e ... C e ...)     appC

(+ 1 (if (+ 2 ■) 3 4))
-> (appC [+, 1]
     (if1C (appC [+ 2] hole [])
           3 4)
     [])
```

2-5

(if (= 2 2) 3 4)

finding a redex
redex — reducible expression

(if $\blacksquare$ 3 4)   (= 2 2)

plug   true

(if true 3 4)

$C[e] = \text{plug}(C, e)$

$\text{plug} : C \times e \Rightarrow e$

plug $\blacksquare$ e = e

plug (ifC C et ef) e = (if C[e] et ef)

plug (appC (b...) C (a...)) e = (b ... C[e] a ...)

2-b/ step v = v

step C[(if false et ef)] = C[ef]

step C[(if ~~v #~~ false et ef)] = C[et]

step C[(p v ...)] = C[δ(p, v ...)]

~~step e = e', then step C[e] = C[e']~~

2-7 When are two programs "the same"?

$$1+1 = 1+1$$
$$= 1 + (2-1)$$

two programs (x, y) are the same as
$$\forall C . \ \text{eval } C[x] = \text{eval } C[y]$$

~~obsersta~~ observational equivalence