Semilattice
- a domain of values V, i.e. the domain is a set of possible values
- a distinguished "top" "T" value
- a binary operator "meet" "^" (V, V) -> V

1. x ^ x = x // idempotent
2. x ^ y = y ^ x // commutative
3. (x ^ y) ^ z = x ^ (y ^ z) // associative

4. T needs to be such that T ^ x = x
Optionally, a semilattice may have a "bottom" value, where bot ^ x = bot

Partial order
- a binary relation "<=" on a set X such that

1. x <= x // reflexive
2. x <= y and y <= x implies x = y // anti-symmetric
3. x <= y and y <= z implies x <= z // transitive

Over the integers
- x <= y iff x | y
- NOT 5 <= 7 or 7 <= 5

For a lattice
- x <= y iff x ^ y = x

#2 assume x <= y and y <= x
x ^ y = x
y ^ x = y

Greatest lower bound
- given a partial order and x, y in our domain
- if g = glb(x, y)

1. g <= x
2. g <= y
3. if z <= x and z <= y, then z <= g

For lattices, x ^ y = glb(x, y)
- set g = x ^ y

#1
g ^ x
(x ^ y) ^ x
x ^ (y ^ x)
x ^ (x ^ y)
(x ^ x) ^ y
x ^ y
g
So g <= x

#3
assume z <= x and z <= y
z ^ x = z
z ^ y = z

z ^ g
z ^ (x ^ y)
(z ^ x) ^ y
z ^ y
z
So z <= g

Monotonicity
A function f: X -> X, given a partial order, is monotonic if
- x <= y implies f(x) <= f(y)

A dataflow analysis framework is:
- direction D which is "forwards" or "backwards"
- a semillatice V with top value T and meet operator ^
- a family of transfer functions { f_s } where f_s: V -> V
  and is monotonic with respect to the partial order on lattices
- boundary conditions (initial values)

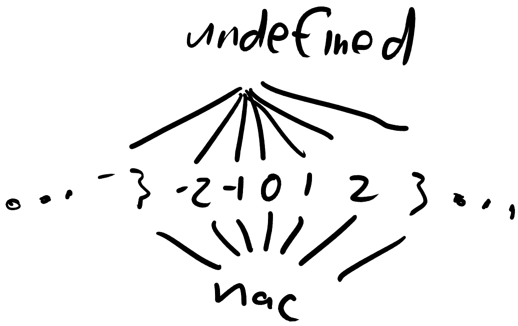
Available expressions
- D = forwards
- V = { all possible sets of expressions }
- ^ = intersection
- T = { all expressions }

Live variables
- D = backwards
- V = { all possible sets of variables }
- ^ = union
- T = {}

Forwards pseudocode
```
OUT[entry] = // starting value
for each node B other than entry, OUT[B] = T
while any changes to any OUT {
    for each node B other than entry {
        IN[B] = meet_{P a predecessor of B} OUT[P]
        OUT[B] = f_B(IN[B])
```

Backwards pseudocode
```
IN[exit] = // starting value
for each node B other than exit, IN[B] = T
while any changes to any IN {
    for each node B other than exit {
        OUT[B] = meet_{S a successor of B} IN[S]
        IN[B] = f_B(OUT[B])
```

Constant propagation
- forwards analysis
- V = { 32 bit integers } union { undefined, not-a-constant }
- Top is "undefined"
- meet(x, y) = x if x == y (x, y are integers)
  = nac if x != y (x, y are integers)
  = undefined if any x, y are undefined
  = nac if x or y are nac (and not undefined)
- f_s(v) if s is "x = y + z"
  = (y + z, y, z) // if both y and z are integers
  = (undefined, y, z) // if any y and z are undefined
  = (nac, y, z) // otherwise
- f_s(v) if s is "x = c" = c



We have a copy of V for each local variable
(V, V, V)
(x1, x2, x3) ^ (y1, y2, y3)
= (x1 ^ y1, x2 ^ y2, x3 ^ y3)

y = 3 out= undef, 3, undef
z = 4 out= undef, 3, 4
x = y+z out= 7, 3, 4