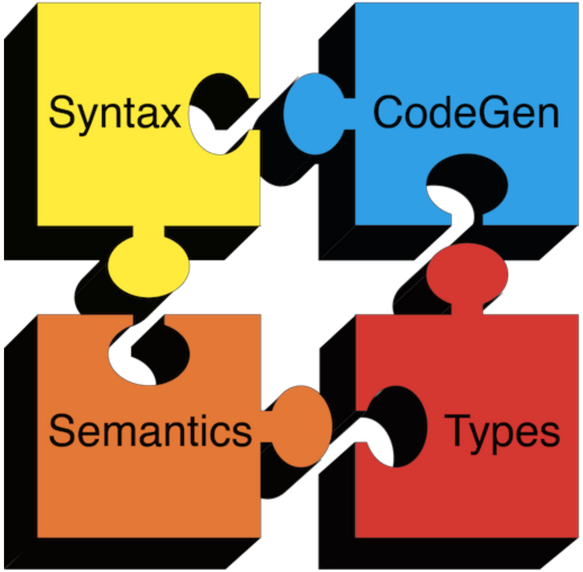


# Compilers

## Orderings

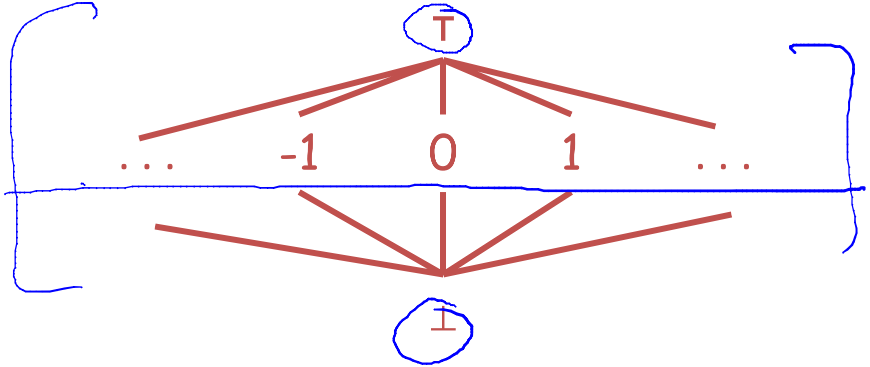
# Orderings

- We can simplify the presentation of the analysis by ordering the values

abstract values

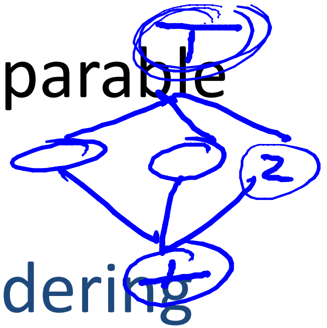
$$\bot < c < \top$$

- Drawing a picture with "lower" values drawn lower, we get



# Orderings

- ⊤ is the greatest value, ⊥ is the least
  - All constants are in between and incomparable



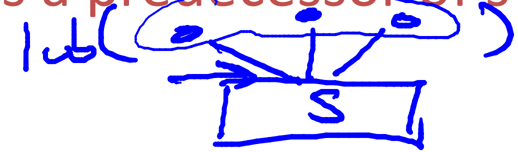
- Let *lub* be the least-upper bound in this ordering

$\mathrm{lub}(\bot, 1) = 1$ $\qquad$ $\mathrm{lub}(1, 2) = \top$

$\mathrm{lub}(\top, \bot) = \top$

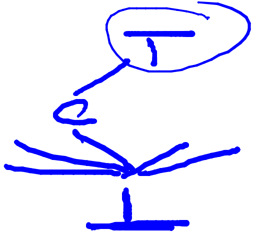
- Rules 1-4 can be written using lub:

C(s, x, in) = lub { C(p, x, out) | p is a predecessor of s }



# Orderings

- Simply saying "repeat until nothing changes" doesn't guarantee that eventually nothing changes



- The use of lub explains why the algorithm terminates
  - Values start as ⊥ and only *increase*
  - ⊥ can change to a constant, and a constant to ⊤
  - Thus, $\underbrace{C(s, x, \overset{\text{in}}{\overset{\text{out}}{\_}})}$ can change at most twice

# Orderings

Thus the constant propagation algorithm is linear in program size

Number of steps =

Number of C(….) values computed * 2 =

Number of program statements * 4