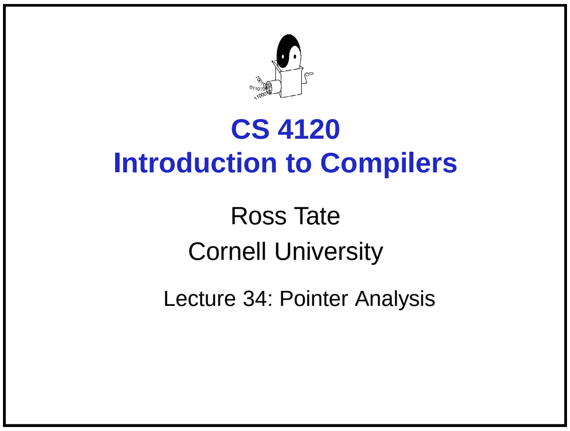
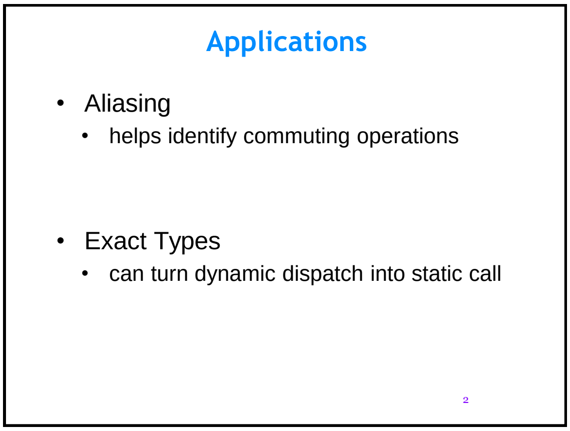


CS 4120
Introduction to Compilers

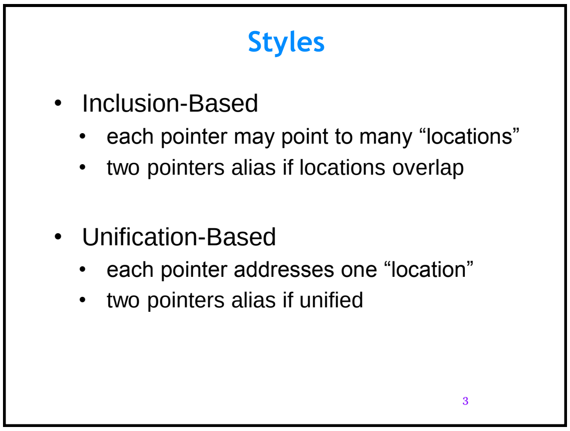
Ross Tate
Cornell University

Lecture 34: Pointer Analysis



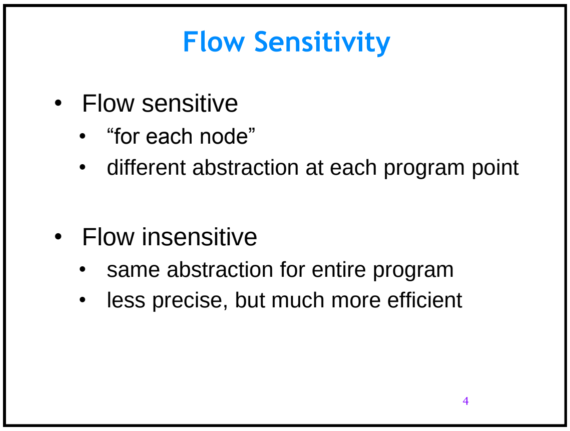
Applications

- Aliasing
  - helps identify commuting operations
- Exact Types
  - can turn dynamic dispatch into static call



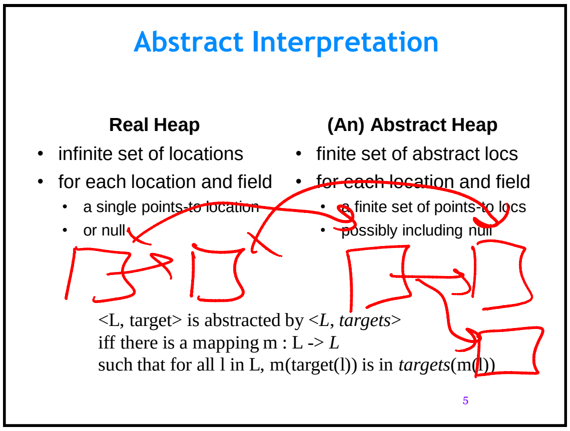
Styles

- Inclusion-Based
  - each pointer may point to many “locations”
  - two pointers alias if locations overlap
- Unification-Based
  - each pointer addresses one “location”
  - two pointers alias if unified



Flow Sensitivity

- Flow sensitive
  - “for each node”
  - different abstraction at each program point
- Flow insensitive
  - same abstraction for entire program
  - less precise, but much more efficient



Abstract Interpretation

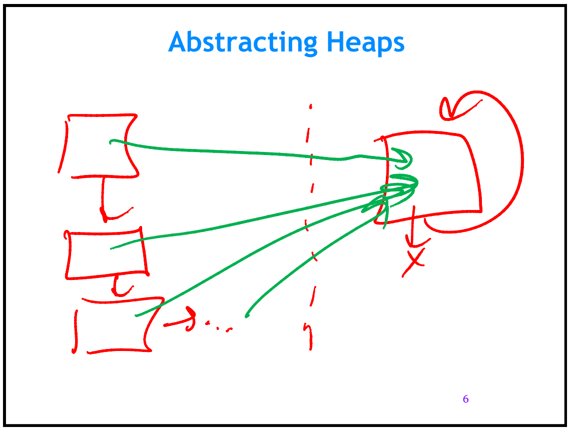
Real Heap

- infinite set of locations
- for each location and field
  - a single points-to location
  - or null

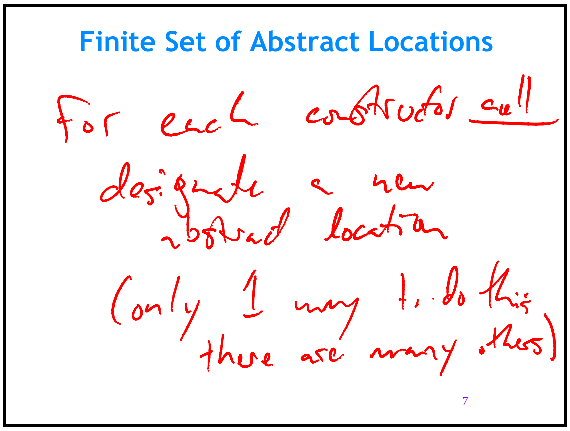
(An) Abstract Heap

- finite set of abstract locs
- for each location and field
  - a finite set of points-to locs
  - possibly including null

<L, target> is abstracted by <*L*, *targets*>
iff there is a mapping m : L -> *L*
such that for all l in L, m(target(l)) is in *targets*(m(l))



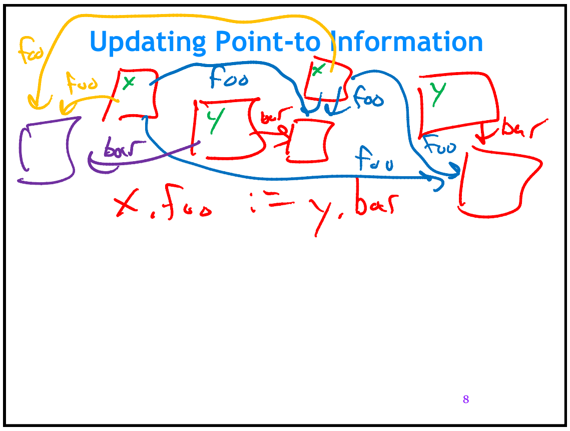
Abstracting Heaps



Finite Set of Abstract Locations

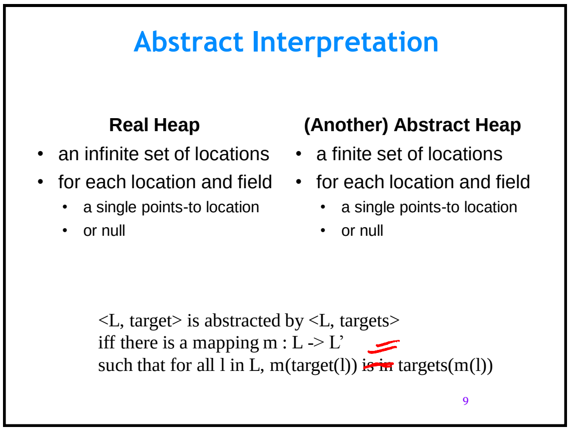
For each constructor call designate a new abstract location

(only 1 way to do this there are many others)



Updating Point-to Information

x.foo := y.bar



Abstract Interpretation

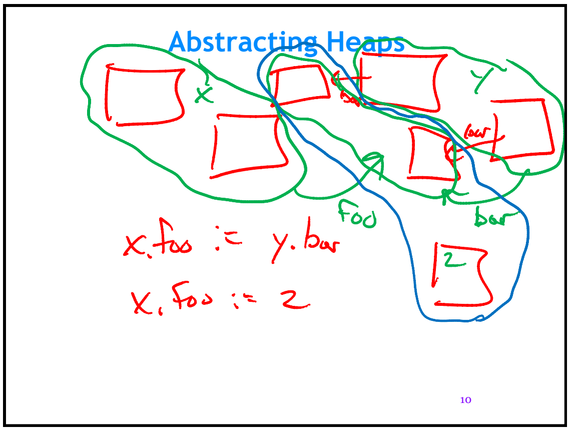
Real Heap

- an infinite set of locations
- for each location and field
  - a single points-to location
  - or null

(Another) Abstract Heap

- a finite set of locations
- for each location and field
  - a single points-to location
  - or null

<L, target> is abstracted by <L, targets>
iff there is a mapping m : L -> L'
such that for all l in L, m(target(l)) ~~is in~~ = targets(m(l))



Abstracting Heaps

x.foo := y.bar

x.foo := z