Constant Propagation

A More Complex Semilattice
A Nondistributive Framework

The Point

- Instead of doing constant folding by RD's, we can maintain information about what constant, if any, a variable has at each point.
- An interesting example of a DF framework not of the gen-kill type.
- A simple version of static type analysis.

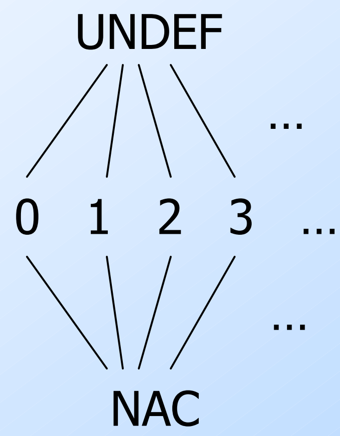
Domain of Values

- The set of values propagated is the set of mappings from variables to values of their type.
- Example: [x→5, s→ "cat", y → UNDEF, z → NAC]
  - UNDEF = "We don't yet know anything."
  - NAC = "Not a constant" = we know too much for any constant to satisfy."

The Semilattice

- A *product lattice*, one component for each variable.
- Each component lattice consists of:
  1. UNDEF (the top element).
  2. NAC (the bottom element).
  3. All values from a type, e.g., integers, strings.

Picture



The Meet Operation

- The diagram represents $\leq$. That is:
  1. Any constant $\leq$ UNDEF.
  2. NAC $\leq$ any constant.
- Equivalently, for any constants x and y:
  1. UNDEF $\wedge$ x = x.
  2. NAC $\wedge$ x = NAC.
  3. NAC $\wedge$ UNDEF = NAC.
  4. $x \wedge x = x$ but $x \wedge y =$ NAC if $x \neq y$.

The Product Lattice

- Call each of the lattices just described a *diamond lattice*.
- The lattices we use are products of diamond lattices.
- For the product $D_1*D_2*\dots*D_n$, the values are $[v_1, v_2,\dots, v_n]$, where each v_i is in D_i.

Meet in Product Lattices

- $[v_1, v_2, \ldots, v_n] \wedge [w_1, w_2, \ldots, w_n] =$ $[v_1 \wedge w_1, v_2 \wedge w_2, \ldots, v_n \wedge w_n] =$ *componentwise meet*.
- In terms of $\leq$:

$$[v_1, v_2, \ldots, v_n] \leq [w_1, w_2, \ldots, w_n]$$

if and only if $v_i \leq w_i$ for all i.

Intuitive Meaning

1. If variable x is mapped to UNDEF (i.e., in the product-lattice value, the component for x is UNDEF), then we do not know anything about x.
2. If x is mapped to constant c, then we only know of paths where x has value c.
3. If x is mapped to NAC, we know about paths where x has different values.

Product-Lattice Values as Mappings

- Think of a lattice element as a mapping from variables to values {UNDEF, NAC, constants}.
- Lattice element is m, and m(x) is the value to which m maps variable x.

Transfer Functions --- (1)

- Transfer functions map lattice elements to lattice elements.
- Suppose m is the variable->constant mapping just before a statement `x = y+z`.
- Let f(m) = m' be the transfer function associated with `x = y+z`.

Transfer Functions --- (2)

- If m(y) = c and m(z) = d, then m'(x) = c+d.
- If m(y) = NAC or m(z) = NAC, then m'(x) = NAC.
- Otherwise, if m(y) = UNDEF or m(z) = UNDEF, then m'(x) = UNDEF.
- m'(w) = m(w) for all w other than x.

Transfer Functions --- (3)

- Similar rules for other types of statements (see text).
- For a block, compose the transfer functions of the individual statements.

Iterative Algorithm

- It's a plain-ol' Forward iteration, with the meet and transfer functions as given.
- The framework is monotone and has bounded depth, so it converges to a safe solution.

Finite Depth

- The value of any IN or OUT can only decrease.
  - Verify from transfer functions (monotonicity).
- Values are finite-length vectors, and each component can only decrease twice.
  - From UNDEF to a constant to NAC.
- If no IN or OUT decreases in any component in a round, we stop.

Monotonicity --- (1)

- Need to show $m \leq n$ implies $f(m) \leq f(n)$.
- Show for function f associated with a single statement.
- Composition of monotone functions is monotone.
- That’s enough to show monotonicity for all possible transfer functions.

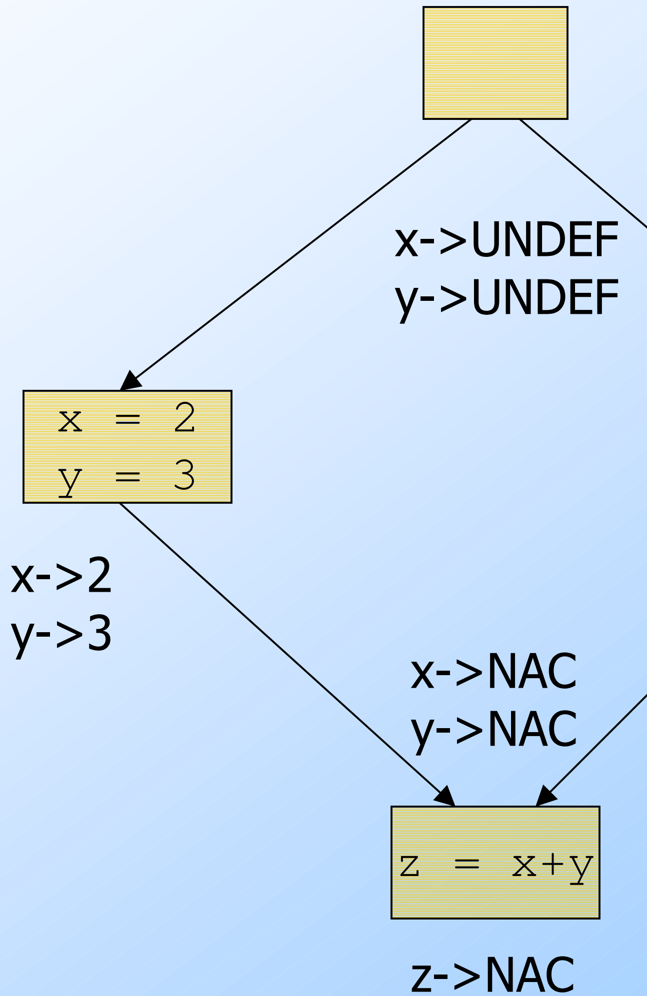
Monotonicity --- (2)

- One case: let f be the function associated with `x = y+z`.
- One subcase: m(y) = c; m(z) = d; n(y) = c; n(z) = UNDEF; m(w) = n(w) otherwise. Thus, m $\leq$ n.
- Then (f(m))(x) = c+d and (f(n))(x) = UNDEF.
- Thus (f(m))(w) $\leq$ (f(n))(w) for all w.

Nondistributivity

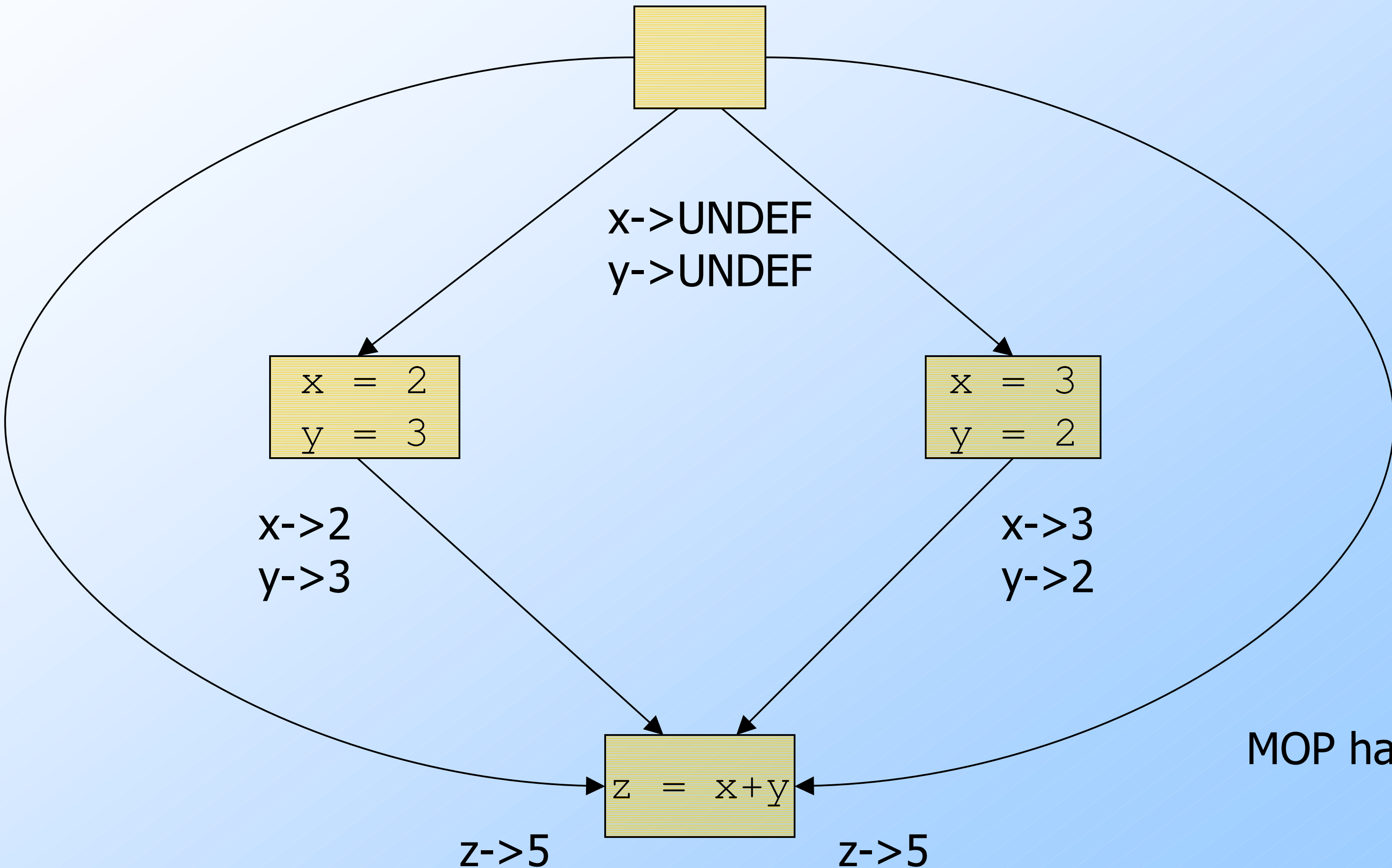
- First example of a framework that is not distributive.
- Thus, iterative solution is not the MOP.
- We'll show an example where MFP appears to include impossible paths.

Example: Nondistributivity



Iterative solution finds z is "not a constant."

Example: Nondistributivity --- (2)



MOP has z = 5.

Example: Nondistributivity --- (3)

- We observe that MFP differs from the MOP solution.
- That proves the framework is not distributive.
  - Because every distributive framework has MFP = MOP.