Lattice-Theoretic Data Flow Analysis Framework

Goals

- provide a single, formal model that describes all DFAs
- formalize notions of “safe”, “conservative”, “optimistic”
- place precise bounds on time complexity of DF analysis
- enable connecting analysis to underlying semantics for correctness proofs

Plan:

- define **domain** of program properties computed by DFA, organize domain elements as a **lattice**
- define flow functions, merge function over this domain using lattice operations
- benefit from lattice theory in attacking above issues

History: Kildall [POPL 73], Kam & Ullman [JACM 76]

Lattices

Define lattice $D = (S, \leq)$:

- S is a set of elements of the lattice
- $\leq$ is a binary relation over elements of S

Required properties of $\leq$:

- $\leq$ induces a **partial order** over S
  - reflexive, transitive, & anti-symmetric
- every pair of elements of S has
  a unique **greatest lower bound** (a.k.a. meet) and
  a unique **least upper bound** (a.k.a. join)

Height of D =
longest path through partial order from greatest to least

- infinite lattice can have finite height (but infinite width)

Top (T) = unique element of S that’s greatest, if exists
Bottom ($\perp$) = unique element of S that’s least, if exists

Lattice models in data flow analysis

Model information by elements of a lattice

- top = best case info
- bottom = worst case info
- merge function = g.l.b. (meet) on lattice elements
- if $a \leq b$, then a is a conservative approximation to b

Reaching definitions:

- elements:
- $\leq$:
  - meet:
  - top:
  - bottom:

Reaching constants:

- elements:
- $\leq$:
  - meet:
  - top:
  - bottom:

Tuples of lattices

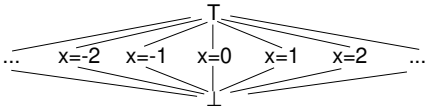
Often helpful to break down a complex lattice into a tuple of lattices, one per variable being analyzed

Formally: $D_T = <S_T, \leq_T> = \text{Tuple}(D = <S, \leq>)$

- $S_T = S \times S \times ... \times S$
  - element of tuple domain is a tuple of elements from each variable’s domain
  - i^{th} component of tuple is info about i^{th} variable
- $<..., d_{1i}, ...> \leq_T <..., d_{2i}, ...> \equiv d_{1i} \leq d_{2i}, \forall i$
  - i.e. pointwise ordering
  - meet: pointwise meet
  - top: tuple of tops
  - bottom: tuple of bottoms
- $\text{height}(D_T) = N * \text{height}(D)$

E.g. reaching constants

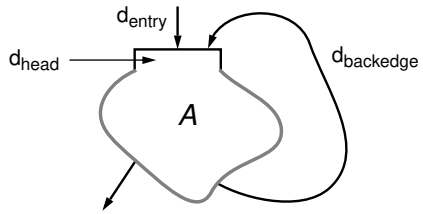
- lattice for single variable is 3-level lattice:



- whole problem is tuple of individual lattices

Analysis of loops in lattice model

Consider:



Assume A(d) computes info at backedge given d, info at head

Want solution to constraints:

$d_{head} = d_{entry} \cap d_{backedge}$

$d_{backedge} = A(d_{head})$

Let $F(d) = d_{entry} \cap A(d)$

Then want fixpoint of F:

$d_{head} = F(d_{head})$

Iterative analysis in lattice model

Iterative analysis computes fixpoint by iterative approximation:

$F^0 = d_{entry} \cap T$

$F^1 = d_{entry} \cap A(F^0) = F(F^0) = F(d_{entry})$

$F^2 = d_{entry} \cap A(F^1) = F(F^1) = F(F(F^0)) = F(F(d_{entry}))$

. . .

$F^k = d_{entry} \cap A(F^{k-1}) = F(F^{k-1}) = F(F(...(F(d_{entry}))...))$

until

$F^{k+1} = d_{entry} \cap A(F^k) = F(F^k) = F^k$

Is k finite?

If so, how big can it be?

Termination of iterative analysis

In general, k need not be finite

Sufficient conditions for finiteness:

- flow functions (e.g. F) are **monotonic**
- lattice is of finite height

A function F is monotonic iff:

$d_1 \le d_2 \Rightarrow F(d_1) \le F(d_2)$

- for application of DFA, this means that giving a flow function more conservative inputs leads to more conservative outputs, never more optimistic outputs

For monotonic F over domain D, the maximum number of times that F can be applied to itself, starting w/ any element of D, w/o reaching fixpoint is height(D)-1

- start at top of D
- go down one level in lattice each application of F
- eventually must hit fixpoint or bottom (which is guaranteed to be a fixpoint), if D of finite height

Complexity of iterative analysis

How long does iterative analysis take?

`l`: depth of loop nesting

`n`: # of stmts in loop

`t`: time to execute one flow function

`k`: height of lattice