Interprocedural, finite, distributive, subset (IDFS) problems [Reps-Horwitz-Sagiv-95]

K. V. Raghavan

IISc

Summary of Sharir-Pneuli approaches

- Requirements of all three approaches (call-strings, functional, iterative)
  - Transfer functions are monotonic
- Additional requirements of functional approach
  - Functions be represented as data-structures
  - The operations of join, composition, and equality-checking be defined on the representation
- Additional requirements of call-strings and iterative approach
  - L is finite
- What they all compute: LFP in general, JVP for distributive functions.

Separable or gen-kill problems

- Lattice for available expressions analysis for a single expression: $AV \equiv \{\perp, 1, 0\}$.
- Lattice for k expressions?

Separable or gen-kill problems

- Lattice for available expressions analysis for a single expression: $AV \equiv \{\perp, 1, 0\}$.
- Lattice for k expressions? AV^k

Separable or gen-kill problems

- Lattice for available expressions analysis for a single expression: $AV \equiv \{\perp, 1, 0\}$.
- Lattice for k expressions? AV^k
  - Call-strings approach would need a very high bound. Iterative approach would need 3^k "columns".
  - Very expensive
- If we analyze for each expression separately using AV lattice, we would get same results as analyzing all of them together using AV^k lattice.
- This is because available-expressions is a "separable" or "gen-kill" problem.
- Other examples of gen-kill problems: reaching definitions, unitialized variables, live variables.

Definition of gen-kill problem

- Lattice $L = 2^D$, where D is a finite set
- Join is union or intersection
- Transfer functions are distributive over union if join is union, and distributive over *both* union and intersection if join is intersection.
- For each transfer function f and for each $d \in D$,
  $f(\{d\}) = f(\emptyset)$ (i.e., f does not "transmit" d) or
  $f(\{d\}) = f(\emptyset) \cup \{d\}$ (i.e., f "transmits" d).

Alternative definition of gen-kill problem

- Lattice $L = 2^D$, where D is a finite set
- Join is union or intersection
- For each node n, there exist two sets gen_n and $kill_n$, both being subsets of D, such that

$$f_n = \lambda S.(S - kill_n) \cup gen_n$$

Strategy for solving gen-kill problems efficiently

1. Do abstract interpretation separately using lattice $2^{\{d\}}$, for each $d \in D$.
   - gen_n and $kill_n$ of each statement n are restricted to $\{d\}$ (i.e., intersected with $\{d\}$).
2. At any program point p, final JOP is union of JOP's using individual analyses.

Proof outline of separability

- Consider Set 1 of dataflow equations, in which variables range over L. Use given $L \rightarrow L$ transfer functions and join operation to frame the equations, as usual.
- Consider the lattice $V = L_1 \times L_2 \times \ldots L_{|D|}$, where each $L_i = \{\emptyset, \{d_i\}\}$.
  - For each statement n, $f_n^V(v \in V)$ returns a vector whose ith slot is computed from $v[i]$ using gen_n and $kill_n$ both restricted to $\{d_i\}$.
  - Similarly, join on the vectors works pointwise using underlying join of L.
- Consider Set 2 of dataflow equations, in which variables range over elements of V, using transfer functions and join operation mentioned above.
- There is a natural one-to-one correspondence between elements of L and elements of V. $l \in L$ and $v \in V$ correspond iff l is equal to union of all components of v.

Proof outline – continued

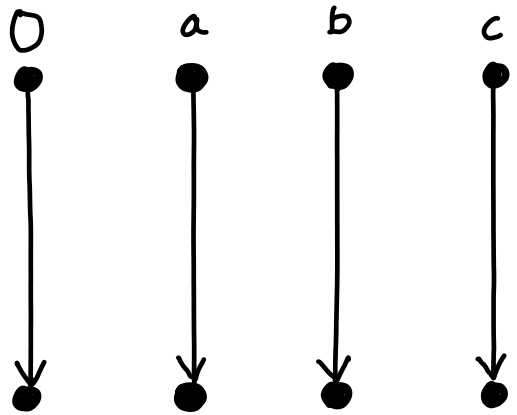
- Let $l \in L$ and $v \in V$ be corresponding. It can be proved that $f(l)$ and $f(v)$ are corresponding, for any transfer function f. Also, if $l_1 \in L$ corresponds with $v_1 \in V$ and $l_2 \in L$ corresponds with $v_2 \in V$, then $l_1 \sqcup l_2$ corresponds with $v_1 \sqcup v_2$.
- It can be proved that for any fix point of the Set 1 equations, there is a fix point of Set 2 equations such that these two fix points correspond to each other as per the correspondence mentioned above. The converse, from fix points of Set 2 to fix points of Set 1 also holds.
- Therefore, the lfp's of the two sets of equations are in correspondence with each other.
- Finally, due to the pointwise nature of all transfer functions and the join operation in Set 2, it is clear that Set 2 equations can be solved one slot at a time.

Requirements of [RHS95] approach

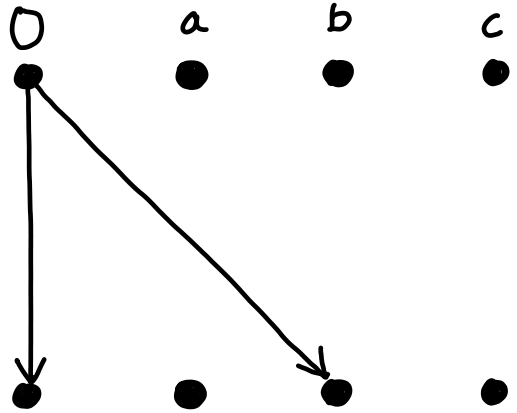
- The abstract lattice $L = 2^D$, where D is a finite set
- Transfer functions are distributive (approach is undefined for non-distributive functions)
- Join (in the L lattice) is union or intersection.
  - Paper focuses on union problems. Every intersection problem has a complement problem whose join is union.
- Gen-kill's restriction on the transfer functions is not present.
- Therefore, every gen-kill problem is also an IDFS (Inter-procedural distributive finite subset) problem.
- Examples of IDFS problems that are *not* Gen/Kill problems: Truly live variables, Possibly uninitialized variables.
- The RHS algorithm computes JVP for IDFS frameworks.

Representing dataflow functions

Identity function
$f = \lambda S.S$



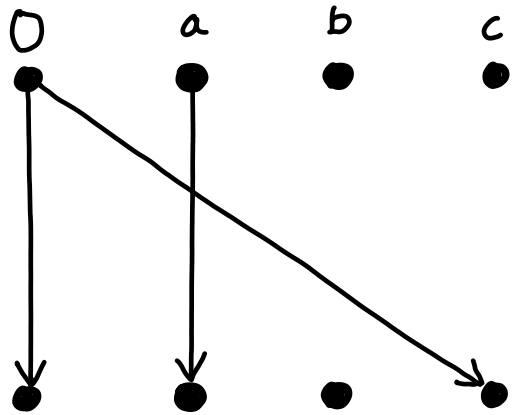
Constant function
$f = \lambda S.\{b\}$



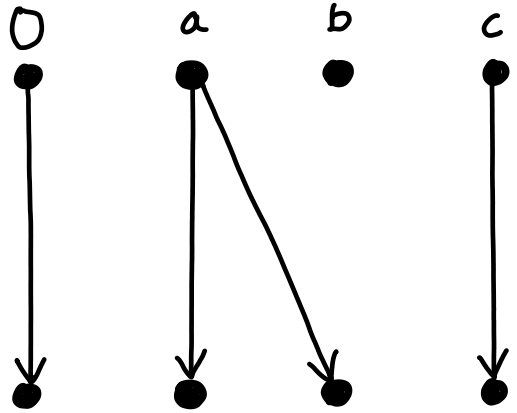
(Credit: Prof. Reps)

Representing dataflow functions - II

Gen/Kill function
$f = \lambda S.(S - \{b\}) \cup \{c\}$

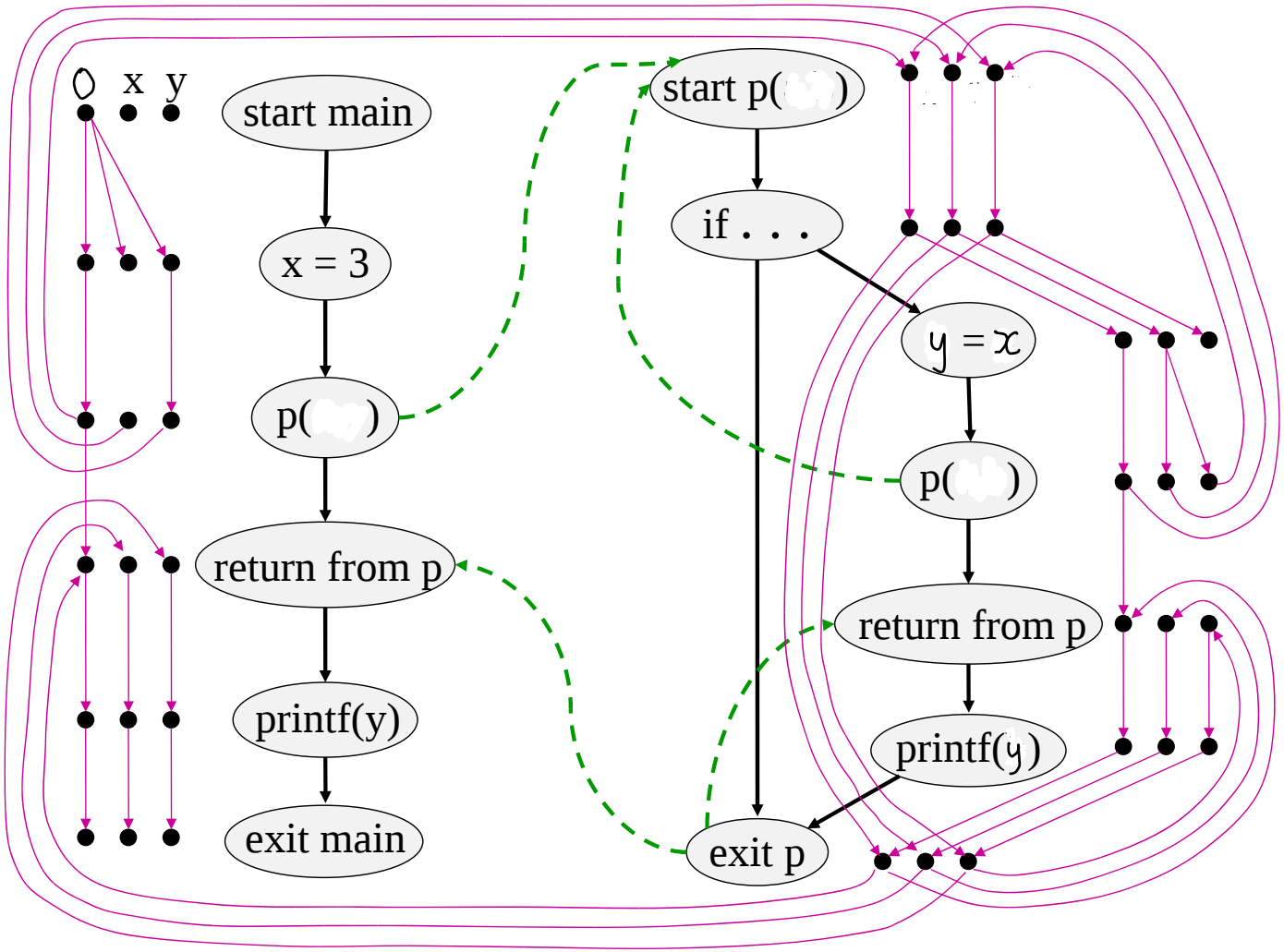


Non-Gen/Kill function
$f = \lambda S.$if $a \in S$
then $S \cup \{b\}$
else $S - \{b\}$



For formal details: See Definition 3.1 in paper (Credit: Prof. Reps)

An example exploded graph for possibly-uninitialized variables using RHS approach



Here, initial value is $\{x, y\}$

Algorithm

Tabulation algorithm, iteratively adds path edges, uses dynamic programming, using rules described in Figures 3 and 4 in the paper.

- Every path edge begins at entry point of a procedure, and ends at some point in the procedure.
- First path edge added: $< r_{main}, \mathbf{0} > \rightarrow < r_{main}, \mathbf{0} >$.
- Path edge $< r_{procOf(N)}, d_1 > \rightarrow < N, d_2 > \Rightarrow$
  - there is an inter-procedurally valid path from $< r_{main}, \mathbf{0} >$ to $< r_{procOf(N)}, d_1 >$, *and*
  - there is an inter-procedurally valid and complete path from $< r_{procOf(N)}, d_1 >$ to $< N, d_2 >$.

What the algorithm computes

- Intuitively, there is a path edge from $< r_{procOf(N)}, q >$ to $< N, d >$ iff
  - $q \in D$ and d is in the JVP at point N whenever q is in the JVP at point $r_{procOf(N)}$, *or*
  - q is $\mathbf{0}$ and d is in the JVP at point N whenever control enters procedure $procOf(N)$
- Theorem 3.8: The JVP at point N with initial value $\mathbf{0}$ includes the element $d \in D$ iff a path edge ends at (N, d); i.e., iff there is a *inter-procedurally valid path* from $< start_{main}, \mathbf{0} >$ to $< N, d >$.

In other words, they have reduced dataflow analysis to graph reachability (along valid paths).

Efficiency

Complexity

- $O(ED^3)$ for general IDFS problems
- $O(ED)$ for gen-kill problems; comparable to best known algorithm specialized for gen-kill problems.

How it compares to iterative approach

- Plain iterative would need time proportional to $|L|$, which is $2^{|D|}$.
- It's possible to produce a variant of iterative approach that works with values from D, instead of values from L. Still, its complexity is likely to be worse than $O(ED^3)$.