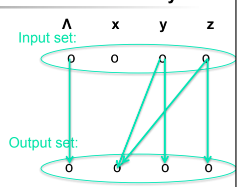
IFDS and CFL-Reachability (optional slides)

IFDS Context Sensitivity

- **I**nterprocedural, **F**inite, **D**istributive, **S**ubset (IFDS) problems
- Allows for efficient computation of summary transfer functions. Converts problem into Context-Free-Language (CFL)-Reachability
  - Can reduce monotone problem into the IFDS problem, but with loss of precisions
- Reading: Thomas Reps, Susan Horwitz and Mooly Sagiv, "Precise, Interprocedural Dataflow Analysis via Graph Reachability, POPL'95

Efficient Encoding of Transfer Functions

E.g., f_j at j: x = y + z
Taint Analysis



- Finite set of dataflow facts **D**
  - E.g., all variables **{x, y, z}**
- Transfer functions **f: $2^D \to 2^D$**
- Edge **Λ** → d means $d \in f(\emptyset)$
  - I.e., **d** is generated
- Edge **d1→d2** means $d2 \notin f(\emptyset)$ and $d2 \in f(S)$ if $d1 \in S$
  - I.e., **d1** in **S** leads to **d2** in **f(S)**
- Edge **Λ** → **Λ** always there

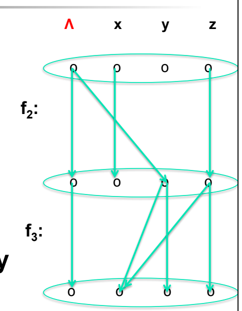
What Can Be Encoded. Taint Analysis

1. z = 5

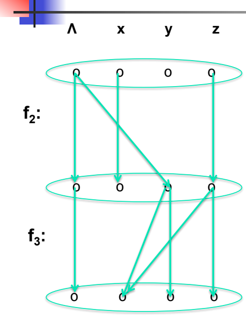
2. y = "tainted" value

3. x = y + z

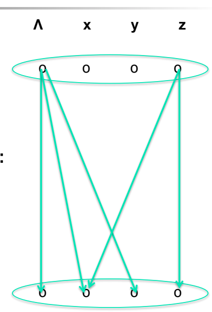
The paths from top **Λ** to **x** and to **y** entail that **x** and **y** are tainted at exit from 3.



Efficient Computation of Function Composition!



$f_3 \circ f_2$:

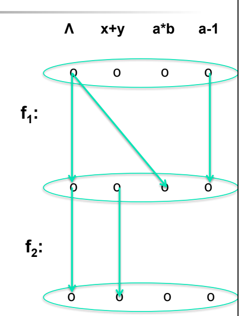


What Can Be Encoded. All Bit-Vector Problems!

1. x = a*b

2. a = a – 1

- Add edges from **Λ** to facts being generated (e.g., **a*b**)
- Add in-out edges to facts being preserved (e.g., **a-1**)



What Cannot Be Encoded

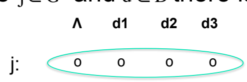
- Monotone functions cannot be encoded
  - E.g., constant propagation, points-to analysis
- Points-to analysis, distributive subset?
  - $\mathbf{f_{p=*q}}$: **p→x** in $\mathbf{f_{p=*q}(S)}$ if **q→y** in **S** AND **y→x** in **S**
  - Can encode disjunctions but not conjunctions
- Large class of problems falls under IFDS
- Monotone problems can be reduced into IFDS with loss of precision

Big Picture, Why Does It Matter

- We can compose transfer functions within a procedure **p** and compute the summary transfer function $\mathbf{\Phi_p}$!
- Precisely: Computes the MORP solution!
- Efficiently: **O(ED³)**
  - **E** is the number of intraprocedural edges across all procedures in ICFG

Exploded Supergraph G#

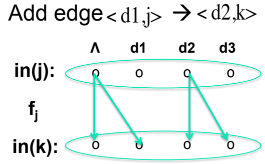
- Let **G*** be the ICFG, which Reps et al. call the supergraph
- First, define the nodes of **G#**
- For each node $j \in G^*$ there is node $<j,\Lambda> \in G\#$
- For each node $j \in G^*$ and $d \in D$ there is node $<j,d> \in G\#$



- Represents the **in(j)**

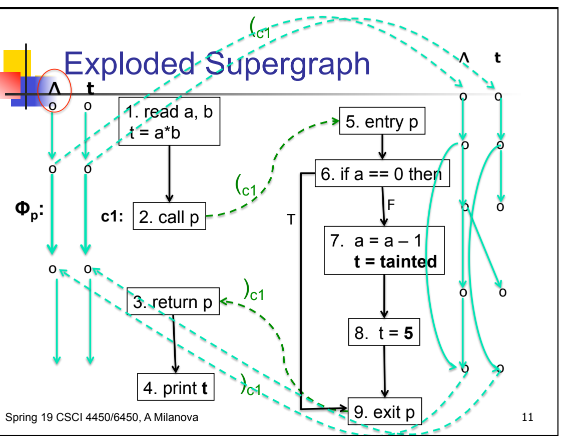
Exploded Supergraph G#

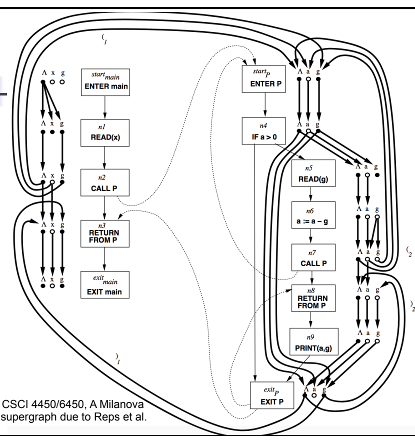
- Next, add edges to **G#**
- For each **k** in successors of **j**
  - Add edge $<j,\Lambda> \rightarrow <k,\Lambda>$ to **G#**
  - Add edge $<j,\Lambda> \rightarrow <k,d>$ if $d \in f_j(\varnothing)$
  - Add edge $<d1,j> \rightarrow <d2,k>$ if $d2 \notin f_j(\varnothing)$ and $d2 \in f_j(in(j))$ if $d1 \in in(j)$



- Represent (encode) transfer function $\mathbf{f_j}$

Exploded Supergraph





Exploded supergraph due to Reps et al.

Exploded Supergraph G#

- One can think about IFDS in terms of Sharir and Pnueli's functional approach
- … or in terms of graph reachability: IFDS reduces the standard dataflow problem to a reachability problem in G#
  - Path from **<1,Λ>** to **<j,d>** means that **d** reaches **j**
  - More precisely, it is a CFL-reachability (Context-Free-Language reachability) problem: "Is there a path from **<1,Λ>** to **<j,d>** whose edges form a string in the language of realizable paths?"
  - Gives rise to on-demand approaches

IFDS Conclusion

- Key idea is encoding of transfer functions $\mathbf{f_j}$
  - Allows for efficient computation of **summary transfer functions** $\mathbf{\Phi_p}$
  - Reduces to CFL-reachability problem on G#
- IFDS is defined for forward **may** problems. Forward **must** problems can be expressed as complement
- Real-world analysis problems
  - Soot has a built-in IFDS framework
  - Some taint analyses for Android use IFDS