IFDS and CFL-Reachability (optional slides)

IFDS Context Sensitivity
- Interprocedural, Finite, Distributive, Subset (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
- Finite set of dataflow facts $D$
  - E.g., all variables $\{x, y, z\}$
  - Transfer functions $f: 2^D \rightarrow 2^D$
- Edge $\Lambda \rightarrow d$ means $d \in f(\emptyset)$
  - i.e., $d$ is generated
- Edge $d_1 \rightarrow d_2$ means $d_2 \notin f(\emptyset)$ and $d_2 \in f(S)$ if $d_1 \in S$
  - i.e., $d_1$ in $S$ leads to $d_2$ in $f(S)$
- Edge $\Lambda \rightarrow \Lambda$ always there

What Can Be Encoded.
Taint Analysis
1. $z = 5$
2. $y = "tainted"$ value
3. $x = y + z$

The paths from top $\Lambda$ to $x$ and to $y$ entail that $x$ and $y$ are tainted at exit from 3.

Efficient Computation of Function Composition!

What Can Be Encoded.
All Bit-Vector Problems!
1. $x = a*b$
2. $a = a - 1$
- Add edges from $\Lambda$ 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?
  - \( f_{p=q} : p \rightarrow x \) in \( f_{p=q}(S) \) if \( q \rightarrow y \) in \( S \) AND \( y \rightarrow x \) in \( S \)
- Can encode disjunctions but not conjunctions

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 \( \Phi_p \)
- Precisely: Computes the MORP solution!
- Efficiently: \( O(ED^2) \)
  - \( E \) 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 \( <j,A> \in G^# \)
  - For each node \( j \in G^* \) and \( d \in D \) there is node \( <j,d> \in G^# \)
- Represents the \( \text{in}(j) \)

Next, add edges to \( G^# \)

- For each \( k \) in successors of \( j \)
  - Add edge \( <j,A> \rightarrow <k,A> \) to \( G^# \)
  - Add edge \( <j,A> \rightarrow <k,d> \) if \( d \in f_j(\emptyset) \)
  - Add edge \( <d_1,p> \rightarrow <d_2,k> \) if \( d_2 \notin f_j(\emptyset) \) and \( d_2 \in f_j(\text{in}(j)) \) if \( d_1 \in \text{in}(j) \)

Represent (encode) transfer function \( f_j \)

Exploded Supergraph

1. read a, b
2. call p
3. return p
4. print t
5. entry p
6. if a == 0 then
7.  a = a - 1
8.  t = tainted
9. exit p

Example: Read a, b

5. entry p
6. if a == 0 then
7.  a = a - 1
8.  t = tainted
9. exit p
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 \( <1,\Lambda> \) 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 \( <1,\Lambda> \) 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 \( f_j \)
  - Allows for efficient computation of summary transfer functions \( \Phi_p \)
  - Reduces to CFL-reachability problem on G#
- IFDS is defined for forward \textit{may} problems. Forward \textbf{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