Precise Interprocedural Dataflow Analysis via Graph Reachability

Precise Interprocedural Dataflow Analysis via Graph Reachability
- Introduction
- IFDS
  - Graph
  - Path
  - Same-level valid path
  - Valid path
  - Instance
  - Path function
  - Representation relation
  - Interpretation
  - Composition
- Conversion
  - fact
  - Exploded supergraph
  - Theorem
- Tabulation Algorithm
  - Four functions
  - Edges
  - pseudo-code
  - Time

Introduction

Intraprocedural: "precise" means "meet-over-all-paths"

Interprocedural: "precise" means "meet-over-all-**valid**-paths"

The paper provides a polynomial-time algorithm for finding precise solutions to a general class of interprocedural dataflow analysis problems. In this problem, the set of dataflow facts `D` is a finite set and the dataflow functions distribute over the meet operator(union or intersection)

So we call it `interprocedural, finite, distribute, subset (IFDS) problems`.

IFDS

Graph

A program is represented using a directed graph $G^* = (N^*, E^*)$, where G^* is called a super graph.

G^* consists of a collection of flow graphs G_1, G_2,...(one for each procedural) and G_{main} (for the main procedural of the program)

Each flowgraph `Gp` has a unique start node s_p and a unique exit node e_p.

The procedural call is represented by two nodes: a `call node` c and a `return-site node` r.

In each G_i, there are ordinary intraprocedural edges that connects the nodes of the individual flowgraphs.

G^* has three edges,

- An intraprocedural `call-to-return-site` edge from c to r.
- An interprocedural `call-to-start` edge from c to the start node of the called procedural
- An interprocedural `exit-to-return-site` edge from the exit node of the called procedural to r.

For `call-to-return-site` and `exit-to-return-site` edges, they permit the information about local variables that holds at the call site to be combined with the information about the global variables that holds at the end of the called procedural.

Path

Set each call node in G^* with a unique index i.

For each c_i, we label c_i's outgoing `call-to-start` edge by the symbol $(_i$ and the incoming `exit-to-return-site` edge of the corresponding return-site node by the symbol $)_i$.

Same-level valid path

For each pair of nodes m, n in the same procedure, a path from m to n is a `same-level valid path` **iff** the sequence of labeled edges in the path is a string in the language of balanced parentheses generated from nonterminal `matched` by the grammar:

$$matched \ \rightarrow \ (_i \ matched \ )_i \ matched \ \mid \ \epsilon$$

It will be used to capture the transmission of effects from m to n, where m and n are in the same procedure, via some sequence of execution steps.

Valid path

For each pair of nodes m, n in supergraph G^*, a path from m to n is a `valid` path **iff** the sequence of labeled edges in the path is the string in the language generated from nonterminal `valid` in the grammar:

$$valid \ \rightarrow \ valid \ (_i \ matched \ \mid \ matched$$

The valid path from s_{main} to n will be used to capture the transmission of effects from s_{main} to n, via some sequence of execution steps.

Instance

An instance IP of an `IFDS` problem is a five-tuple $IP = (G^*, D, F, M, \sqcap)$, where

- G^* is a supergraph
- D is a finite set for the variables...
- $F \subseteq 2^D \rightarrow 2^D$ is a set of distributive functions(If f is a distributive function, and set $D = D_1 \cup D_2 \cup ... \cup D_k$, then $f(D) = f(D_1) \cup f(D_2) \cup ... \cup f(D_k)$)
- $M : E^* \rightarrow F$ is a map from G^*'s edges to dataflow functions
- $\sqcap$ is meet operator, which is either `union` or `intersect`

Path function

Let $IP = (G^*, D, F, M, \sqcap)$ be an `IFDS` problem instance, and let $q = [e_1, e_2, \ldots, e_j]$ be a non-empty path in G^*.

The `path function` that corresponds to `q` $: pf_q = f_j \diamond f_{j-1} \diamond \ldots \diamond f_2 \diamond f_1$, where for all i such that $1 \leq i \leq j$, $f_i = M(e_i)$. Also, notation $\diamond$ means the composition of two functions. The `path function` for an empty path is the identity function $\lambda x. x$.

We denote the set of all valid paths from m to n by $IVP(m, n)$.

The `meet-over-all-valid-paths` solution to IP consists of the collection of values MVP_n defined as

$MVP_n = \sqcap_{q \in IVP(s_{main}, n)} pf_q(\top)$ for each $n \in N^*$.

Representation relation

The `representation relation` of `f` , $R_f \subseteq (D \cup 0) \times (D \cup 0)$ is a binary relation defined as:

$R_f = \{(0,0)\} \cup \{(0,y) \mid y \in f(\emptyset)\} \cup \{(x,y) \mid y \in f(\{x\})\ and\ y \notin f(\emptyset)\}$.

Interpretation

Given a relation $R \subseteq (D \cup \{0\}) \times (D \cup \{0\})$, its `interpretation` $[R] : 2^D \to 2^D$ is the function defined as

$[R] = \lambda X. (\{y \mid \exists x \in X$ such that $(x,y) \in R\} \cup \{y \mid (0,y) \in R\}) - \{0\}$

Then, it is obvious that $[R_f] = f$

Composition

Given two relations $R_f \subseteq S \times S$ and $R_g \subseteq S \times S,$ their composition $R_f; R_g \subseteq S \times S$ is defined as

$R_f; R_g = \{(x,y) \in S \times S \mid \exists z \in S\ s.t.\ (x,z) \in R_f\ and\ (z,y) \in R_g\}$

It is obvious that, for all $f, g \in 2^D \to 2^D, [R_f; R_g] = g \diamond f$

So the distributive functions in $2^D \to 2^D$ can be represented by a graph(relation) .

Also, we have $f_j \diamond f_{j-1} \diamond \ldots \diamond f_2 \diamond f_1 = [R_{f_1}; R_{f_2}; \ldots; R_{f_j}]$

Conversion

How to convert `IFDS` to `realizable-path` graph reachability problems?

For each instance IP in `IFDS` problem, we construct a graph $G^{\#}_{IP}$ and an instance of `realizable-path` graph reachability problem. The edges of IP corresponds to the representation relations of the dataflow functions on the edges of G^*.

fact

Dataflow-fact d holds at supergraph node n **iff** there is a "realizable path" from a distinguished node in $G^{\#}_{IP}$ to the node in $G^{\#}_{IP}$ that represents the fact d at node n.

Also, $\emptyset$ holds at the start of procedure $main$.

Exploded supergraph

Let $IP = (G^*, D, F, M, \cup)$ be an `IFDS` problem instance, we define exploded supergraph as follows:

$G^{\#}_{IP} = (N^{\#}, E^{\#})$, where

$N^{\#} = N^* \times (D \cup \{0\})$

$E^{\#} = \{< m, d_1 > \rightarrow < n, d_2 > | \; edge \; (m, n) \in E^* \; and \; (d_1, d_2) \in R_{M(m,n)}\}$

Theorem

Let $G^{\#}_{IP} = (N^{\#}, E^{\#})$ be the exploded supergraph for `IFDS` problem instance $IP = (G^*, D, F, M, \cup)$, and let n be a program point in N^*. Then $d \in MVP_n$ **iff** there is a realizable path in graph $G^{\#}_{IP}$ from node $< s_{main}, 0 >$ to node $< n, d >$.

(Note that $MVP_n = \sqcap_{q \in IVP(s_{main}, n)} pf_q(\top)$)

Tabulation Algorithm

`Tabulation Algorithm` is an efficient algorithm for `Realizable-Path Reachability` Problem, which is based on `dynamic programming`. This function is not `path-sensitive` because it does not iterate over every possible execution path???

Four functions

- `returnSite`: call node $\rightarrow$ return site node
- `procOf`: node $\rightarrow$ the name of its enclosing procedure
- `calledProc`: call node $\rightarrow$ the name of called procedure
- `caller`: procedure name $\rightarrow$ the set of call nodes that call to that procedure

Edges

- `PathEdge`: a set to record the existence of `path edges`, which represents the suffix of the `same-level realizable paths` in graph $G^{\#}_{IP}$.
- `SummaryEdge`: a set to record the existence of `summary edges`, which represent `same-level realizable paths` that run from `nodes` of the form $< n, d_1 >$ to $< returnSite(n), d_2 >$, where $n \in Call$. So summary edges represent the partial information about how the dataflow value after a call depends on the dataflow value before the call.

The `Tabulation Algorithm` is a worklist algorithm that accumulates sets of `PathEdge` and `SummaryEdge`. At last. we can check the path edge and get the elements in `MVP`

pseudo-code

Input: `IFDS` instance $IP = (G^*, D, F, M, \cup)$

Init:

// Initialize it with a 0-length same-level realizable path

get exploded graph $G^{\#}_{IP} = (N^{\#}, E^{\#})$

set $PathEdge = \{< s_{main}, 0 > \rightarrow < s_{main}, 0 >\}$

set $SummaryEdge = \emptyset$

set $WorkList = \{<s_{main}, 0> \to <s_{main}, 0>\}$

Main:

// deduces the existence of additional path edges and summary edges

ForwardTabulate();

// get the elements in MVP_n

for each $n \in N^*$

$X_n = \{d_2 \in D \mid \exists d_1 \in (D \cup \{0\})\ such\ that < s_{procOf(n)}, d_1 > \to < n, d_2 > \in PathEdge\}$

// to propagate edge e into $PathEdge$ and $WorkList$

Propagate(e):

if $e \notin PathEdge$

Insert e into $PathEdge$ and $WorkList$

// worklist algorithm based on dynamic programming

ForwardTabulate():

while $WorkList \neq \emptyset$

select and remove an edge $< s_p, d_1 > \to < n, d_2 >$ from $WorkList$

// 1. consider a call node

if n is a `Call node`

// search in a new procedure $p = calledProc(n)$

Propagate($< s_{calledProc(n)}, d_3 > \to < s_{calledProc(n)}, d_3 >$)

// Just like propagate a summary edge???

Propagate($< s_p, d_1 > \to < returnSite(n), d_3 >$)

// 2. consider an exit node for current procedure p

else if n is an exit node of procedure p

// Insert a summary edge???

$c = callers(p)$

Insert $< c, d_4 > \to < returnSite(c), d_5 >$ into $SummaryEdge$

// Restart the processing that finds the same-level realizable paths ???

for each d_3 such that $< s_{procOf(c)}, d_3 > \to < c, d_4 > \in PathEdge$

Propagate($< s_{procOf(c)}, d_3 > \to < c, d_4 >$)

// 3. consider other cases

else

// propagate further

for each $< m, d_3 >$ such that $< n, d_2 > \rightarrow < m, d_3 >$

Propagate($< s_p, d_1 > \rightarrow < m, d_3 >$)

Time

$T = O(ED^3)$, where E is number of super graph edges and D is the size of finite domain.