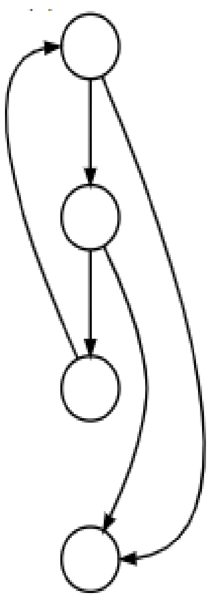


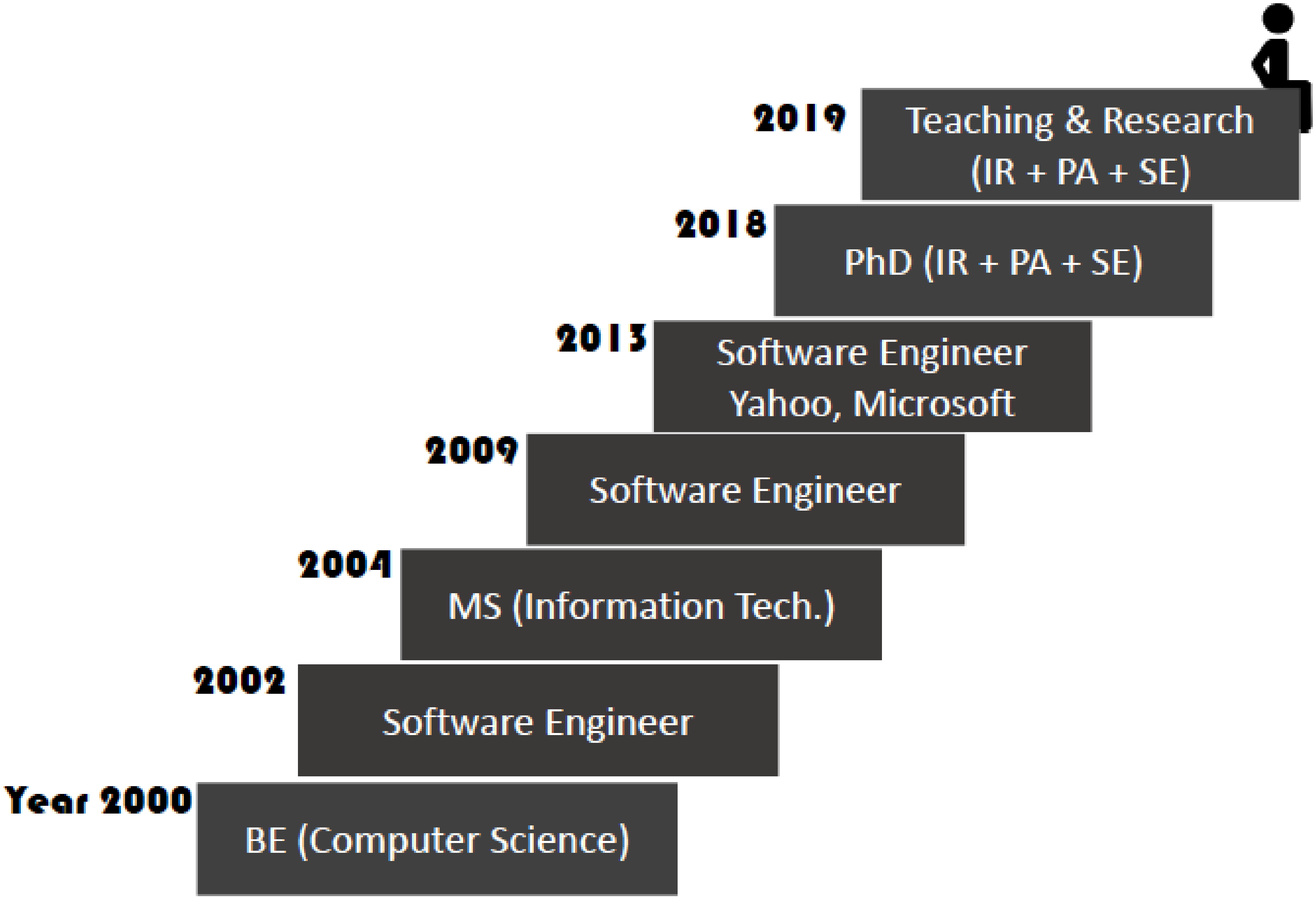
Graphs in Program Analysis

Venkatesh Vinayakarao
Chennai Mathematical Institute
venkateshv@cmi.ac.in
November, 2019

"The supply of grand challenges … shows little sign of drying up."

– Harman and O'Hearn in "Opportunities and Open Problems for Static and Dynamic Program Analysis", Madrid, Spain, 2018.

About Me


2019
Teaching & Research
(IR + PA + SE)
2018
PhD (IR + PA + SE)
2013
Software Engineer
Yahoo, Microsoft
2009
Software Engineer
2004
MS (Information Tech.)
2002
Software Engineer
Year 2000
BE (Computer Science)

Agenda

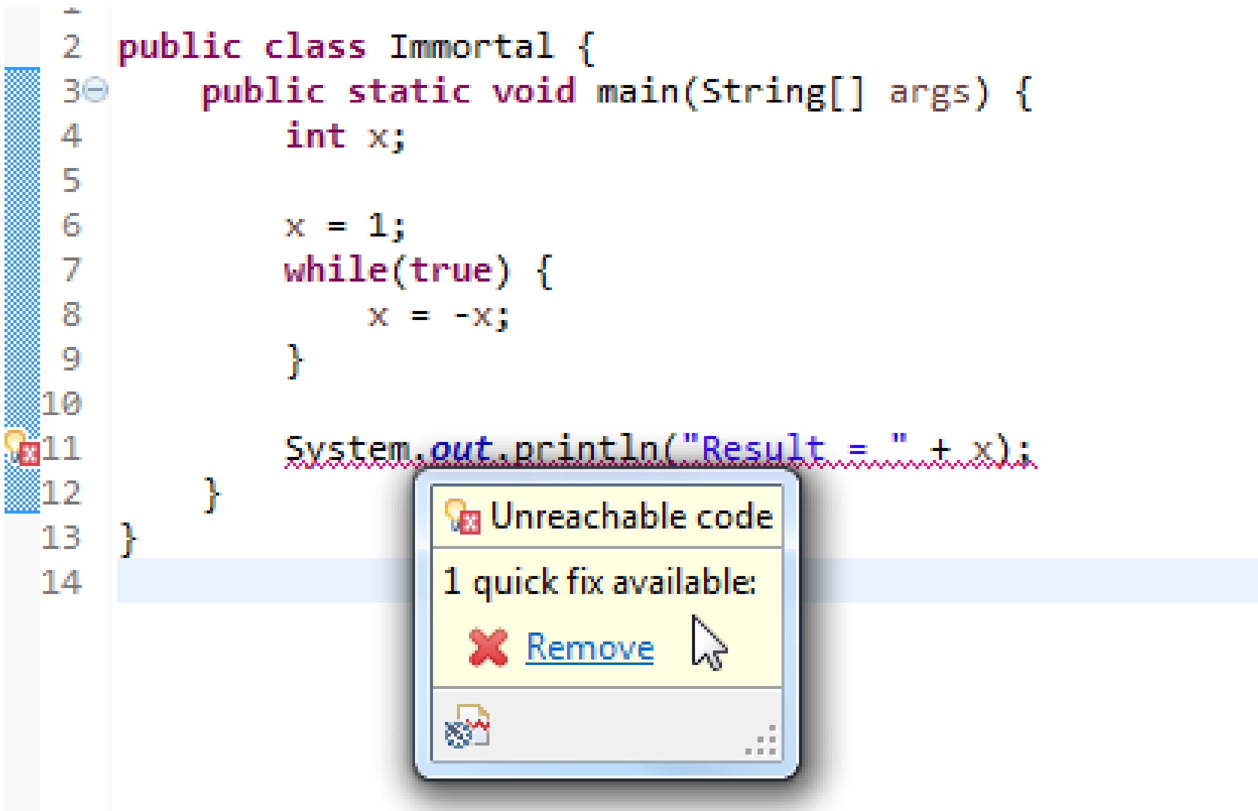
- Program Analysis – A Gentle Introduction
- Why analzye programs?
- A Data Flow Analysis Framework
- Research Trends

Problem in Code!

```
class Immortal {
  public static void main(String[] args) {

    int x;

    x = 1;
    while(true){
         x = -x;
    }

    System.out.println("Result = " + x);
  }
}
```

Any problem in this code?

Built into Eclipse

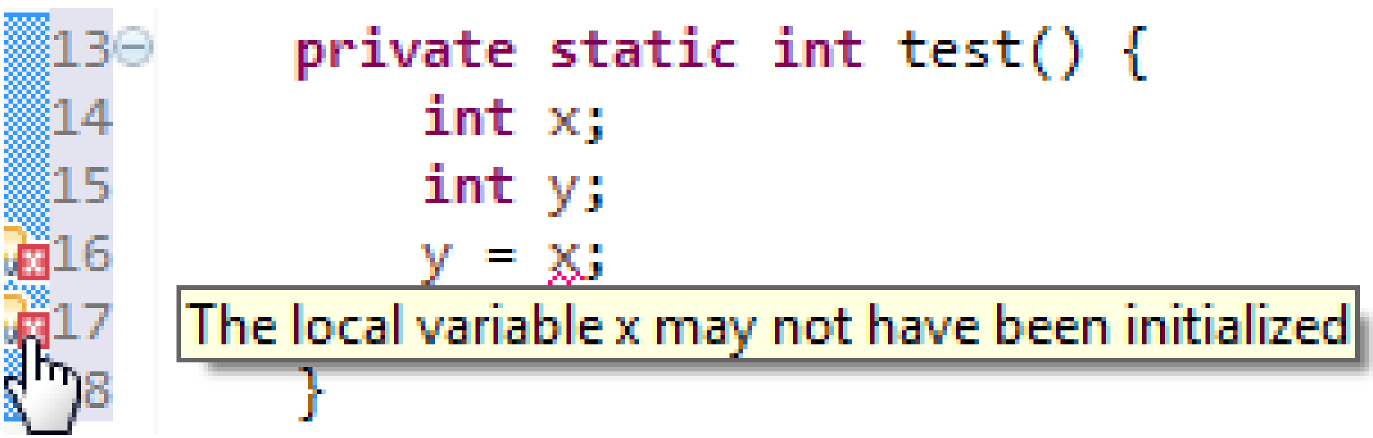


Another Problem Code!

```
private static int test() {
    int x;
    int y;
    y = x;
    return x;
}
```

Any problem in this code?

IDE Catches Some!



Source Code Optimization

- How to optimize this?

```
if (x != 5) x = 5;
```

- Simply,

```
x = 5;
```

http://icps.u-strasbg.fr/~bastoul/local_copies/lee.html

Constant Folding and Propagation

```
int a = 30;
int b = 9 - (a / 5);
int c;

c = b * 4;
if (c > 10) {
  c = c - 10;
}
return c * (60 / a);
```

→

```
int a = 30;
int b = 3;
int c;

c = b * 4;
if (c > 10) {
  c = c - 10;
}
return c * 2;
```

→

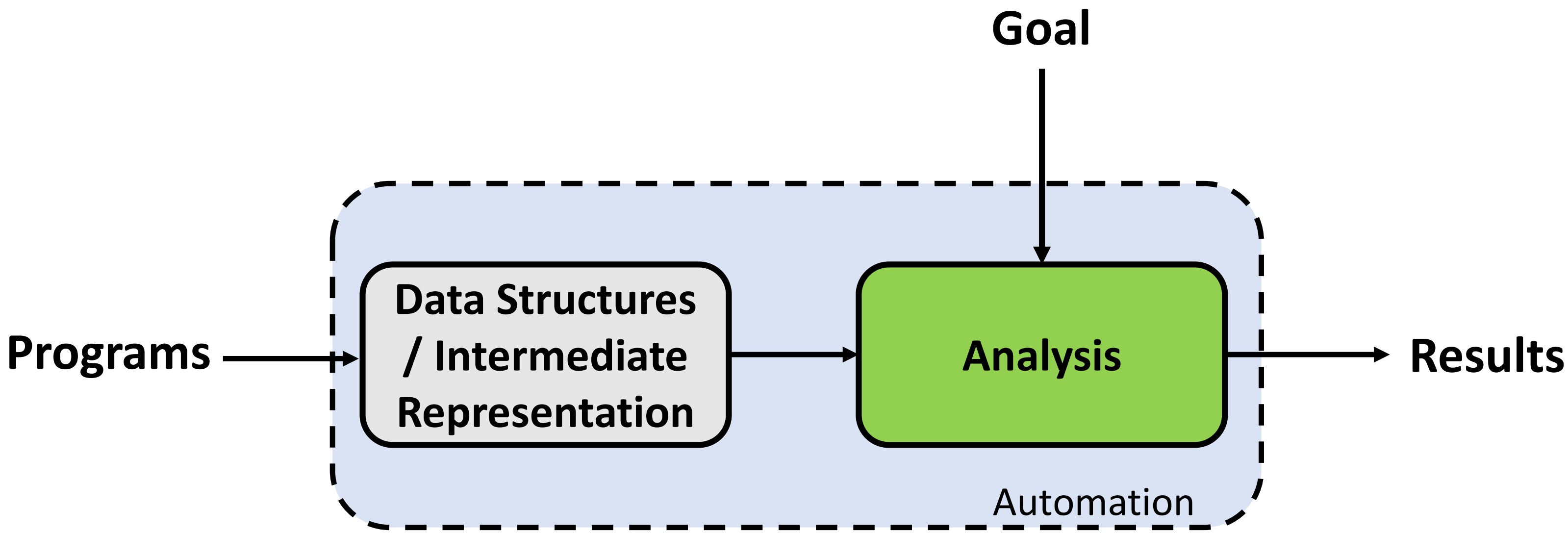
```
int a = 30;
int b = 3;
int c;

c = 12;
if (true) {
  c = 2;
}
return c * 2;
```

→

```
int c;
c = 12;
c = 2;
return c * 2;
```

```
return 4;
```

Program Analysis



Why Study Program Analysis?

Everyone is in a hurry…

Software Reliability: An Issue

For more, visit http://www.cs.tau.ac.il/~nachumd/horror.html

Mars Orbiter Crash

- Primary Cause: Results reported in wrong units
- ”Various officials at NASA have stated that NASA itself was at fault **for failing to make the appropriate checks and tests** that would have caught the discrepancy.”

https://spectrum.ieee.org/aerospace/robotic-exploration/why-the-mars-probe-went-off-course

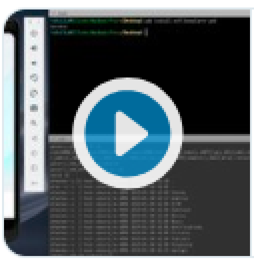
Security Breaches

Aadhaar details leaked after TRAI chief throws breach challenge

Alleged personal details of Indias telecom watchdog chief R.S. Sharma were leaked on Saturday after the TRAI chairman threw a challenge and tweeted his 12-digit Aadhaar asking if it had made him vulnerable to any security risk.

IANS | Updated: July 29, 2018, 08:47 IST

Elliot Alderson @fs0c131y · Jan 30
With more than 100,000,000 downloads @ESFileExplorer is on
famous Android file manager. Bonus: the list of applications in
victim's phone is stored in an unsecured way.



ES File Explorer is leaking the apps installed on yo...
With more than 100,000,000 downloads ES File Explorer is one of the most famous Android file manager. Bonus: the list of applications installed on the tel is...
youtube.com

10 34 87

2:59 AM - 12 Feb 2019

Pinned Tweet
Elliot Alderson @fs0c131y · Feb 12
A new #Aadhaar breach is coming cc @UIDAI 😘

101 462 1.2K

Microsoft Research

Working at Microsoft ⌄ Students and graduates ⌄ Find a job ⌄ Things to do ⌄

Back to search results

RSDE

Looking for an individual that can apply programming language techniques to improve the performance and correctness of software executing in the cloud. The cloud is a major investment for Microsoft, costing us large sums of capital investment and requiring high quality of service guarantees for our customers.

We have already demonstrated through several RiSE projects that applying PL techniques to these problems (including model checking, symbolic execution, semantic abstractions, etc.) we can significantly improve cloud software over the current state of the art. Existing projects in this area include P and P#, Parasail, Uncertain, Network Verification, and Retro. RiSE has been successful in applying foundational reasoning to cloud software problems and leveraging our deep understanding and tool investment (e.g., Z3 SMT solver, Zing model checker, etc.) to create unique and effective solutions. A candidate should have both

Job #	981039
Locations	United States, Redmond (WA)
Job families	Research
Teams	Research

Apply now   Add to job watch list

Empower your futu

MathWorks

Software Engineer - Dynamic Program Analysis

MathWorks · Bangalore, IN

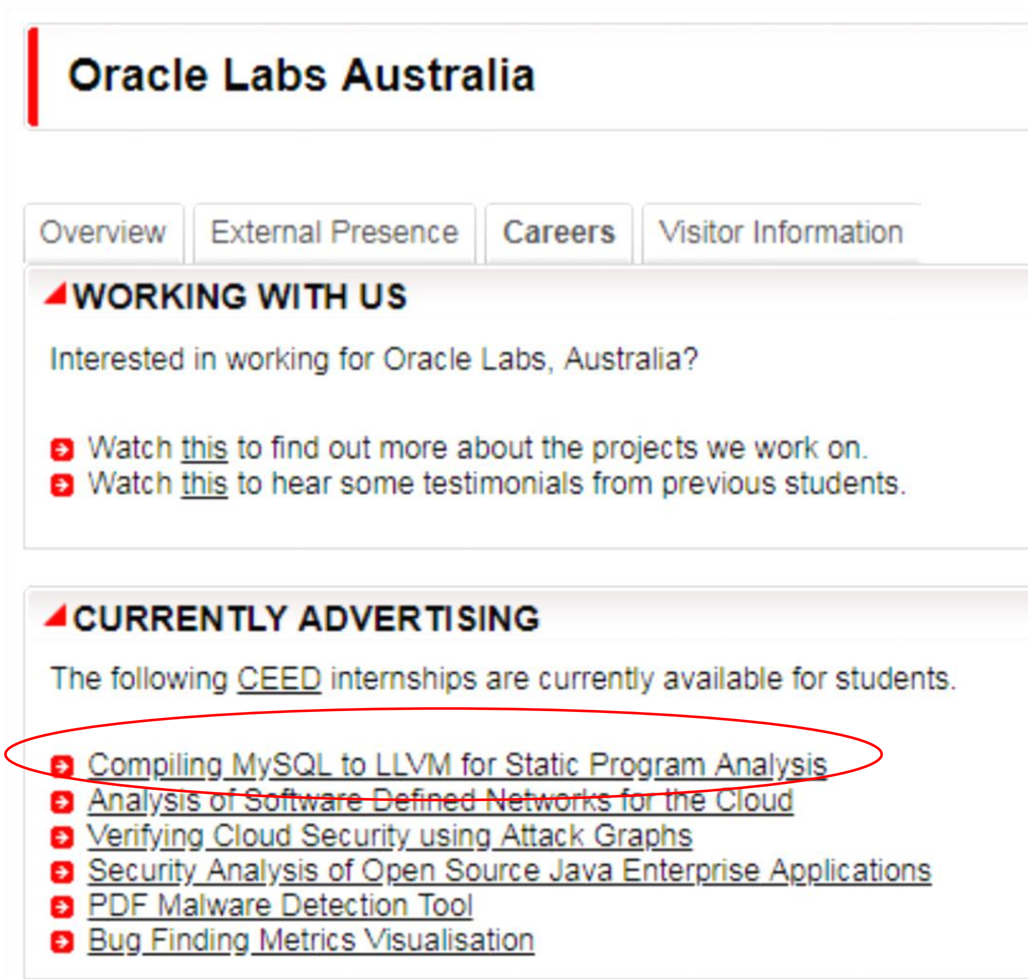
Posted 2 weeks ago · 64 views

Save Apply

who is good at abstract thinking
is a plus. You will join a dynamic
and debugging capabilities to
to learn many of our core
technologies and apply your design and implementation skills to build parts of our product from ground up.

- Design data-structures and algorithms for data-flow analysis of Simulink/Stateflow models and generated code
- Build customer visible UIs for configuring and invoking analysis and transformation engines
- Participate in architecture and design reviews

Oracle Labs



Oracle Labs Australia

Overview | External Presence | Careers | Visitor Information

WORKING WITH US

Interested in working for Oracle Labs, Australia?

- Watch this to find out more about the projects we work on.
- Watch this to hear some testimonials from previous students.

CURRENTLY ADVERTISING

The following CEED internships are currently available for students.

- Compiling MySQL to LLVM for Static Program Analysis
- Analysis of Software Defined Networks for the Cloud
- Verifying Cloud Security using Attack Graphs
- Security Analysis of Open Source Java Enterprise Applications
- PDF Malware Detection Tool
- Bug Finding Metrics Visualisation

IBM IRL

Productive Parallel Programming

Current object-oriented languages have revealed several drawbacks with respect to parallel/concurrent programming at the level of unstructured threads with lock-based synchronization. IBM Research is developing X10, a modern object-oriented programming language designed for high performance with explicit programmer defined parallelism for realizing high productivity programming of parallel computer systems. The key features of X10 include explicit reification of locality in the form of places, support for a partitioned global address space (PGAS) across places, and lightweight activities embodied in async, future, foreach, and ateach constructs which subsume communication and multithreading operations in other languages. Our current focus is on static program analysis (for example, May-Happen-in-Parallel analysis, Bad Place Analysis), compilation for C/C++, debugging for X10, and assessment and semi-automated migration of domain-specific serial code to emerging multi-core architectures, leveraging productive programming models and their variants (such as OpenMP, OpenCL).

More...

CodeSearch - Senior Software Engineer - Program Analysis

Elastic · Yokohama-shi, JP

Posted 2 weeks ago · 207 views

Save Apply

Principal Researcher - Phd/Program Analysis

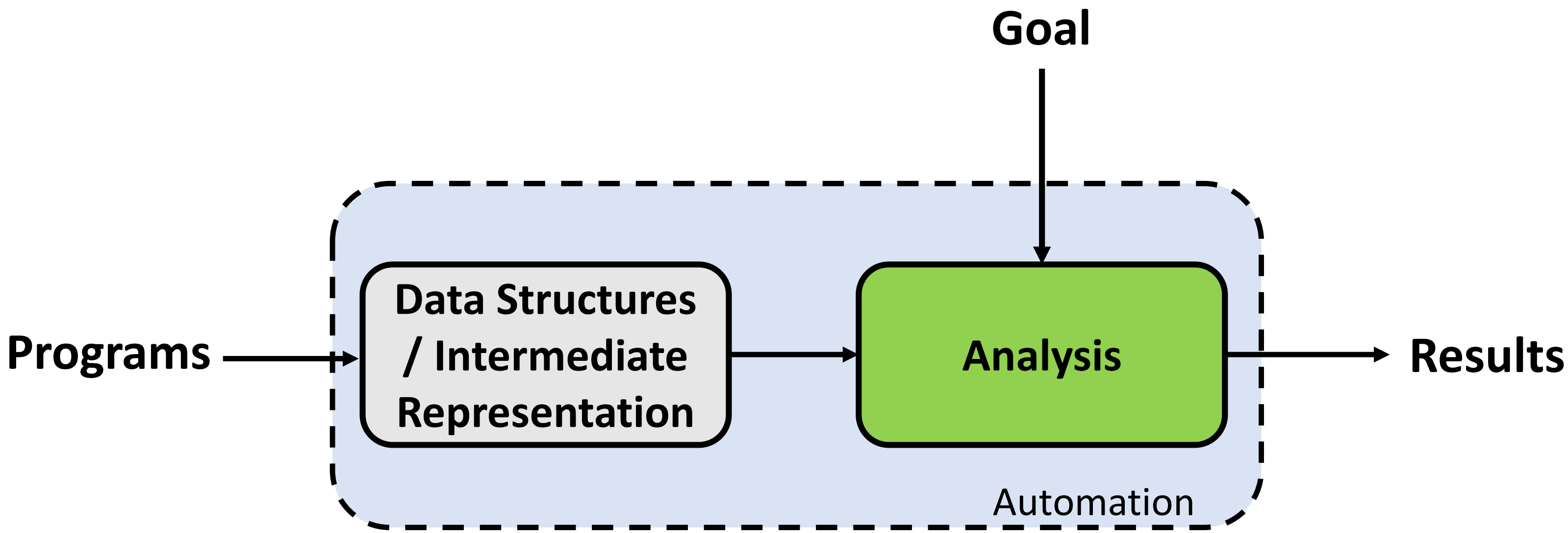
Oracle · Brisbane, Australia

Posted 6 days ago · 79 views

Save Easy Apply

Graphs in Program Analysis

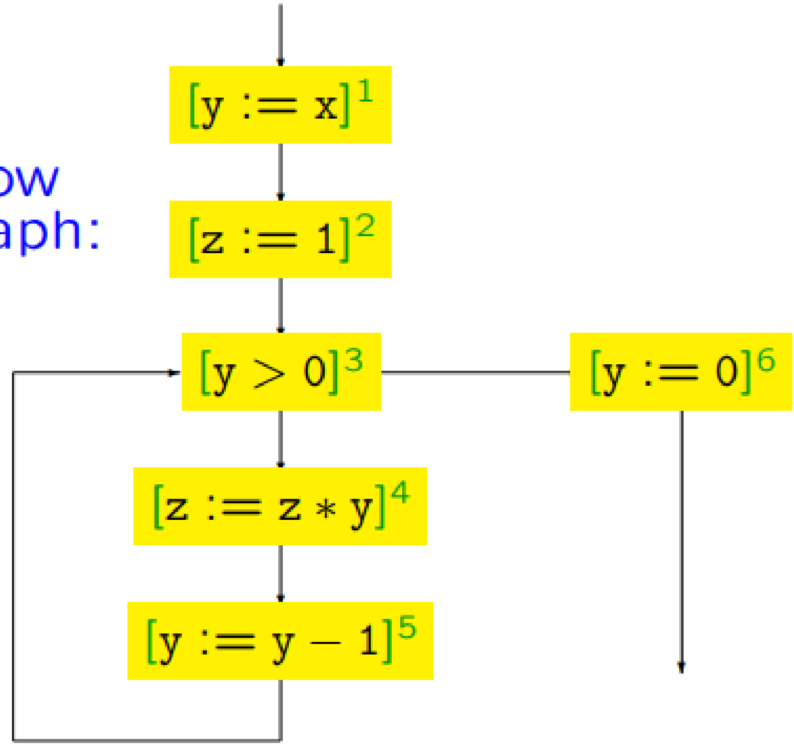
Program Analysis



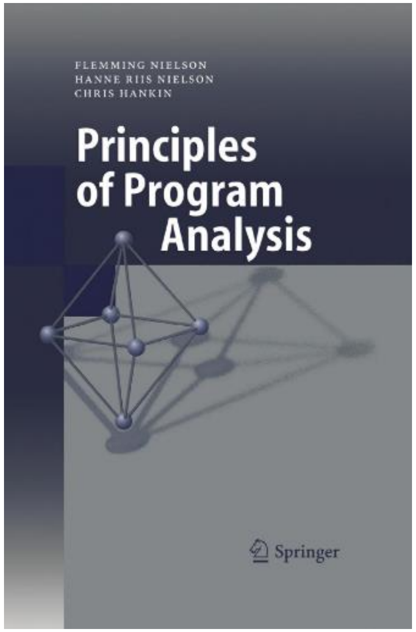
Reaching Definitions (RD) Analysis

```
y = x;
z = 1;
while (y > 0) {
    z = z * y;
    y = y – 1;
}
y = 0
```

Flow graph:



The assignment $[x := a]^{\ell}$ reaches ℓ' if there is an execution where x was last assigned at ℓ. Does $[z := 1]^2$ reach 5?



Data Flow Analysis

The Classic Four!

Reference: Principles of Program Analysis, by Nielson, Nielson and Hankin.

WHILE Language

- Simple Imperative Language
- S refers to Statements, a is an Arithmetic Expression and b is a Boolean Expression

$$a ::= x \mid n \mid a_1 \; op_a \; a_2$$

$$b ::= true \mid false \mid not \; b \mid b_1 \; op_b \; b_2 \mid a_1 \; op_r \; a_2$$

$$S ::= [x := a]^l \mid [skip]^l \mid S_1; S_2 \mid$$

$$\quad if \; [b]^l \; then \; S_1 \; else \; S_2 \mid while \; [b]^l \; do \; S$$

Labeled Programs and Control Flow

$[y := x]^1;$

$[z := 1]^2;$

while $[y > 0]^3$ do

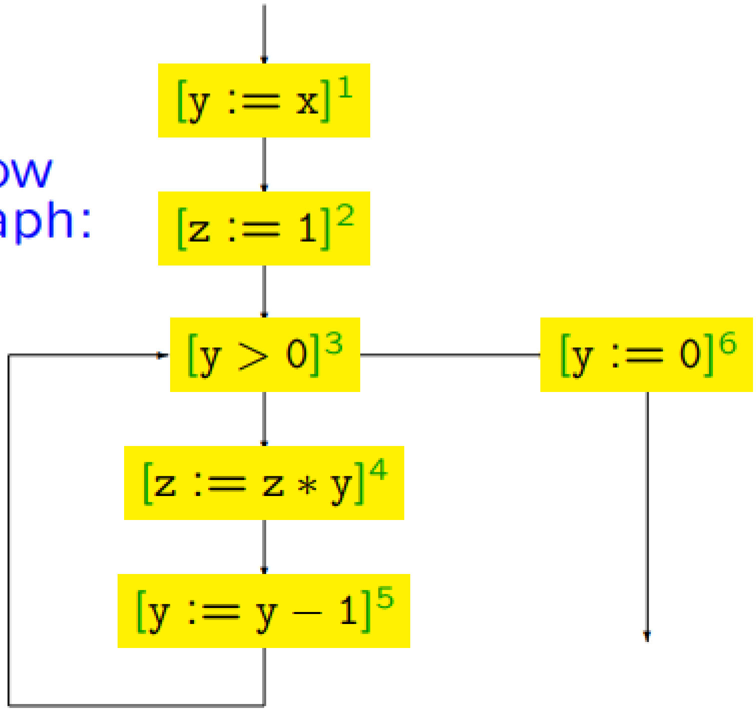
$\quad [z := z * y]^4;$

$\quad [y := y - 1]^5$

od;

$[y := 0]^6$

Flow graph:



Example taken from Principles of Program Analysis, Nielson et al.

Reaching Definitions (RD) Analysis

- The assignment $[x := a]^{\ell}$ reaches ℓ' if there is an execution where x was last assigned at ℓ.

```
[y = x]^1;
[z = 1]^2;
while [(y > 0)]^3 {
    [z = z * y]^4;
    [y = y – 1]^5;
}
[y = 0]^6
```

Does $[z := 1]^2$ reach 5?

RD Analysis

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2		
3		
4		
5		
6		

<u>Labeled</u>
<u>Input Program</u>

```
[y = x]^1;
[z = 1]^2;
while [(y > 0)]^3 {
    [z = z * y]^4;
    [y = y – 1]^5;
}
[y = 0]^6
```

RD Analysis

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2	(x,?),(y,1),(z,?)	(x,?),(y,1),(z,2)
3		
4		
5		
6		

<u>Labeled</u>
<u>Input Program</u>

```
[y = x]^1;
[z = 1]^2;
while [(y > 0)]^3 {
    [z = z * y]^4;
    [y = y – 1]^5;
}
[y = 0]^6
```

RD Analysis

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2	(x,?),(y,1),(z,?)	(x,?),(y,1),(z,2)
3	(x,?),(y,1),(z,2) (z,4),(y,5)	(x,?),(y,1),(z,2),(z,4) ,(y,5)
4		
5		
6		

Labeled Input Program

```
[y = x]1;
[z = 1]2;
while [(y > 0)]3 {
    [z = z * y]4;
    [y = y – 1]5;
}
[y = 0]6
```

RD Analysis

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2	(x,?),(y,1),(z,?)	(x,?),(y,1),(z,2)
3	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,2),(z,4),(y,5)
4	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,4),(y,5)
5	(x,?),(y,1),(z,4),(y,5)	(x,?),(y,5),(z,4)
6	(x,?),(y,1),(z,2),(z,4),(y,5)	(x,?),(y,6),(z,2),(z,4)

<u>Labeled</u>
<u>Input Program</u>

$[y = x]^1$;
$[z = 1]^2$;
while $[(y > 0)]^3$ {
    $[z = z * y]^4$;
    $[y = y – 1]^5$;
}
$[y = 0]^6$

How to Automate?

- We write a system of equations

$$RD_{exit}(1) = (RD_{entry}(1) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,1) \}$$
$$RD_{exit}(2) = (RD_{entry}(2) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,2) \}$$
$$RD_{exit}(3) = RD_{entry}(3)$$
$$RD_{exit}(4) = (RD_{entry}(4) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,4) \}$$
$$RD_{exit}(5) = (RD_{entry}(5) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,5) \}$$
$$RD_{exit}(6) = (RD_{entry}(6) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,6) \}$$

where **Lab** = {1,2,3,4,5,6}

System of Equations…

- Similarly, specify $RD_{entry}(\ell)$ for each line.

$$RD_{entry}(2) = RD_{exit}(1)$$
$$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$$
$$RD_{entry}(4) = RD_{exit}(3)$$
$$RD_{entry}(5) = RD_{exit}(4)$$
$$RD_{entry}(6) = RD_{exit}(3)$$

$$RD_{entry}(1) = \{(x,?),(y,?),(z,?)\}$$

System of Equations…

- 12 Equations with 11 unknowns

Find the least solution

- We have a 12-Tuple, $\overrightarrow{RD}$ = $RD_{entry}(1)$,… ,$RD_{exit}(6)$
- $\overrightarrow{RD}$ = F(RD)

A Simple Iterative Algorithm

$\overrightarrow{RD}$ = (Ø,…. Ø)

j = 0;

while $\overrightarrow{RD} \neq F(RD_1, \cdots, RD_{12})$

do $\overrightarrow{RD} := F(RD_1, \cdots, RD_{12})$

Simple Iteration

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	∅	∅
2	∅	∅
3	∅	∅
4	∅	∅
5	∅	∅
6	∅	∅

F(RD) ➡

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	{(x,?),(y,?),(z,?)}	∅
2	∅	∅
3	∅	∅
4	∅	∅
5	∅	∅
6	∅	∅

F(RD)

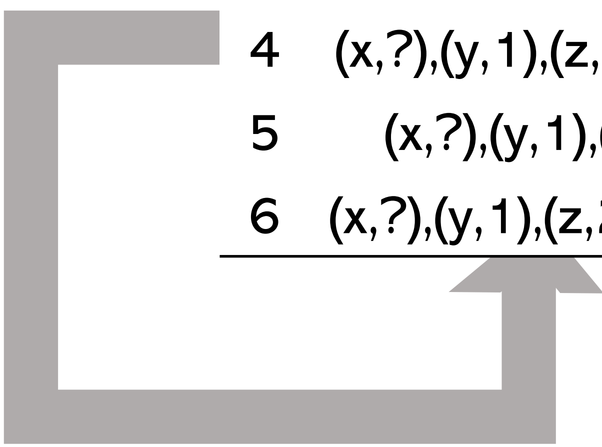
$RD_{entry}(1) = \{(x,?),(y,?),(z,?)\}$
$RD_{entry}(2) = RD_{exit}(1)$
$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$
$RD_{entry}(4) = RD_{exit}(3)$
$RD_{entry}(5) = RD_{exit}(4)$
$RD_{entry}(6) = RD_{exit}(3)$

$RD_{exit}(1) = (RD_{entry}(1) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,1) \}$
$RD_{exit}(2) = (RD_{entry}(2) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,2) \}$
$RD_{exit}(3) = RD_{entry}(3)$
$RD_{exit}(4) = (RD_{entry}(4) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,4) \}$
$RD_{exit}(5) = (RD_{entry}(5) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,5) \}$
$RD_{exit}(6) = (RD_{entry}(6) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,6) \}$

Reaches a Fixed Point

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2	(x,?),(y,1),(z,?)	(x,?),(y,1),(z,2)
3	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,2),(z,4),(y,5)
4	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,4),(y,5)
5	(x,?),(y,1),(z,4),(y,5)	(x,?),(y,5),(z,4)
6	(x,?),(y,1),(z,2),(z,4),(y,5)	(x,?),(y,6),(z,2),(z,4)

F(RD)



The Question

- Does the definition of z in line 2 reach line 5?

$[y := x]^1;$

$[z := 1]^2;$

$\texttt{while}\ [y > 0]^3\ \texttt{do}$

$\quad [z := z * y]^4;$

$\quad [y := y - 1]^5$

$\texttt{od};$

$[y := 0]^6$

Answer: No!
Since, $RD_{entry}(5)$ = (x,?),(y,1),(z,4),(y,5)

There is no (z,2) in it.

The Setup: Some Preliminaries

- *The initial label of a statement*.

init: Stmt -> Lab

$\text{init}([x{:=}a]^\ell) = \ell$
$\text{init}([S_1;S_2]^\ell) = \text{init}(S_1)$
$\text{init}([\text{skip}]^\ell) = \ell$
$\text{init}(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) = \ell$
$\text{init}(\text{while } [b]^\ell \text{ do } S) = \ell$

Setup

- **Final Labels**

final: Stmt -> P(Lab)

$\text{final}([x{:=}a]^\ell) = \{\ell\}$
$\text{final}([S_1;S_2]^\ell) = \text{final}(S_2)$
$\text{final}([\text{skip}]^\ell) = \{\ell\}$
$\text{final}(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) = \text{final}(S_1) \cup \text{final}(S_2)$
$\text{final}(\text{while } [b]^\ell \text{ do } S) = \{\ell\}$

Setup

- We use Blocks to refer to set of statements

blocks: Stmt -> P(Blocks)

$\text{blocks}([x:=a]^\ell) = \{[x:=a]^\ell\}$
$\text{blocks}([S_1;S_2]^\ell) = \text{blocks}(S_1) \cup \text{blocks}(S_2)$
$\text{blocks}([\text{skip}]^\ell) = \{[\text{skip}]^\ell\}$
$\text{blocks}(\text{if } [b]^\ell \text{ then } S_1 \text{ else } S_2) = \{[b]^\ell\} \cup \text{blocks}(S_1) \cup \text{blocks}(S_2)$
$\text{blocks}(\text{while } [b]^\ell \text{ do } S) = \{[b]^\ell\} \cup \text{blocks}(S)$

Setup

- We refer to a statement with a label

labels: Stmt -> P(Lab)

$$\text{labels}(S) = \{ \ell \mid [B]^\ell \in \text{blocks}(S) \}$$
$$\text{init}(S) \in \text{labels}(S)$$
$$\text{final}(S) \subseteq \text{labels}(S)$$

Setup

- The edges of our flow graphs are captured using a flow function.

flow: Stmt -> P(Lab x Lab)

$\text{flow}([x{:=}a]^{\ell}) = \emptyset$

$\text{flow}(S_1;S_2) = \text{flow}(S_1) \cup \text{flow}(S_2) \cup \{(\ell, \text{init}(S_2)) \mid \ell \in \text{final}(S_1)\}$

$\text{flow}([\text{skip}]^{\ell}) = \emptyset$

$\text{flow}(\text{if } [b]^{\ell} \text{ then } S_1 \text{ else } S_2) =$

$\quad \text{flow}(S_1) \cup \text{flow}(S_2) \cup \{(\ell, \text{init}(S_1)), (\ell, \text{init}(S_2))\}$

$\text{flow}(\text{while } [b]^{\ell} \text{ do } S) = \text{flow}(S) \cup \{(\ell, \text{init}(S))\} \cup \{(\ell', \ell) \mid \ell' \in \text{final}(S)\}$

---

flow denotes forward flow here.

Example

- Program power is given below:

$$[z:=1]^1;\ \text{while}\ [x>0]^2\ \text{do}\ ([z:=z*y]^3;\ [x:=x-1]^4)$$

What are $init(power)$, $final(power)$, $labels(power)$ and $flow(power)$?

Example

- Program power is given below:

$$[z:=1]^1; \text{while } [x>0]^2 \text{ do } ([z:=z*y]^3; [x:=x-1]^4)$$

What are $init(power)$, $final(power)$, $labels(power)$ **and** $flow(power)$**?**

$$init(power) = 1$$
$$final(power)=\{2\}$$
$$labels(power) = \{1,2,3,4\}$$
$$flow(power)= \{(1,2),(2,3),(3,4),(4,2)\}$$

Label Consistency Assumption

- All blocks are uniquely labeled.

$[B_1]^\ell, [B_2]^\ell \in \text{blocks}(S) \Rightarrow B_1 = B_2$

Generalizing Data Flow Equations

Recall, RD Equations Were…

$RD_{entry}(1) = \{(x,?),(y,?),(z,?)\}$
$RD_{entry}(2) = RD_{exit}(1)$
$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$
$RD_{entry}(4) = RD_{exit}(3)$
$RD_{entry}(5) = RD_{exit}(4)$
$RD_{entry}(6) = RD_{exit}(3)$

$RD_{exit}(1) = (RD_{entry}(1) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,1) \}$
$RD_{exit}(2) = (RD_{entry}(2) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,2) \}$
$RD_{exit}(3) = RD_{entry}(3)$
$RD_{exit}(4) = (RD_{entry}(4) \setminus \{ (z,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (z,4) \}$
$RD_{exit}(5) = (RD_{entry}(5) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,5) \}$
$RD_{exit}(6) = (RD_{entry}(6) \setminus \{ (y,\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{ (y,6) \}$

Generalizing the Entry and Exit

$$RD_{entry}(\ell) = \begin{cases} \{ (x,?) \mid x \in Var_* \} & \text{if } \ell = init(S_*) \\ \bigcup \{RD_{exit}(\ell') \mid (\ell', \ell) \in flow(S_*)\} & \text{otherwise} \end{cases}$$

$$RD_{exit}(\ell) = (RD_{entry}(\ell) \setminus kill_{RD}(B^\ell)) \cup gen_{RD}(B^\ell)$$
$$\text{where } B^\ell \in blocks(S_*)$$

May Analysis

Forward Analysis

Least Solution Desired

The kill and gen Functions

$kill_{RD}([x{:=}a]^{\ell}) = \{(x,?)\} \cup \{(x, \ell') \mid B^{\ell'} \text{ is an assignment to } x \text{ in } S^*\}$
$kill_{RD}([skip]^{\ell}) = \emptyset$
$kill_{RD}([b]^{\ell}) = \emptyset$

$gen_{RD}([x{:=}a]^{\ell}) = \{(x, \ell)\}$
$gen_{RD}([skip]^{\ell}) = \emptyset$
$gen_{RD}([b]^{\ell}) = \emptyset$

The Kill and gen Sets

ℓ	$\mathbf{kill}_{RD}(\ell)$	$\mathbf{gen}_{RD}(\ell)$
1	{(x,?),(x,1),(x,5)}	{(x,1)}
2	{(y,?),(y,2),(y,4)}	{(y,2)}
3	∅	∅
4	{(y,?),(y,2),(y,4)}	{(y,4)}
5	{(x,?),(x,1),(x,5)}	{(x,5)}

Let us now write the flow equations and solve them to find the reaching definitions.

Labeled Input Program

```
[x = 5]^1;
[y = 1]^2;
while [(x > 1)]^3 {
    [y = x * y]^4;
    [x = x – 1]^5;
}
```

Flow Equations

$RD_{entry}(1) = \{(x,?),(y,?)\}$
$RD_{entry}(2) = RD_{exit}(1)$
$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$
$RD_{entry}(4) = RD_{exit}(3)$
$RD_{entry}(5) = RD_{exit}(4)$

$RD_{exit}(1) = (RD_{entry}(1) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{ (x,1) \}$
$RD_{exit}(2) = (RD_{entry}(2) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{ (y,2) \}$
$RD_{exit}(3) = RD_{entry}(3)$
$RD_{exit}(4) = (RD_{entry}(4) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{ (y,4) \}$
$RD_{exit}(5) = (RD_{entry}(5) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{ (x,5) \}$

Summary

```
[x = 5]¹;
[y = 1]²;
while [(x > 1)]³ {
    [y = x * y]⁴;
    [x = x – 1]⁵;
}
```

Input Program

Data Flow Analysis

ℓ	**$kill_{RD}(\ell)$**	**$gen_{RD}(\ell)$**
1	{(x,?),(x,1),(x,5)}	{(x,1)}
2	{(y,?),(y,2),(y,4)}	{(y,2)}
3	∅	∅
4	{(y,?),(y,2),(y,4)}	{(y,4)}
5	{(x,?),(x,1),(x,5)}	{(x,5)}

$RD_{entry}(1) = \{(x,?),(y,?)\}$
$RD_{entry}(2) = RD_{exit}(1)$
$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$
$RD_{entry}(4) = RD_{exit}(3)$
$RD_{entry}(5) = RD_{exit}(4)$

$RD_{exit}(1) = (RD_{entry}(1) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{ (x,1) \}$
$RD_{exit}(2) = (RD_{entry}(2) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{ (y,2) \}$
$RD_{exit}(3) = RD_{entry}(3)$
$RD_{exit}(4) = (RD_{entry}(4) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{ (y,4) \}$
$RD_{exit}(5) = (RD_{entry}(5) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{ (x,5) \}$

ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	{(x,?),(y,?)}	{(y,?),(x,1)}
2	{(y,?),(x,1)}	{(x,1),(y,2)}
3	{(x,1),(y,2),(y,4),(x,5)}	{(x,1),(y,2),(y,4),(x,5)}
4	{(x,1),(y,2),(y,4),(x,5)}	{(x,1),(y,4),(x,5)}
5	{(x,1),(y,4),(x,5)}	{(y,4),(x,5)}

Analysis Result

Live Variable Analysis

- A variable is live if there is a path from the label to a use of the variable that does not re-define the variable.

```
x = 2;
y = 4;
x = 1;
if (y > x) {
      z = y;
} else {
      z = y * y;
}
x = z;
```

(x,1) is not live at exit.

Useful in Dead code Elimination and register allocation

Summary

```
[x := 2]1;
[y := 4]2;
[x := 1]3;
if [(y > x)]4 {
    [z := y]5;
} else {
    [z := y * y]6;
}
[x := z]7;
```

Input Program

Data Flow Analysis

ℓ	$kill_{LV}(\ell)$	$gen_{LV}(\ell)$
1	{x}	Ø
2	{y}	Ø
3	{x}	Ø
4	Ø	{x,y}
5	{z}	{y}
6	{z}	{y}
7	{x}	{z}

$LV_{entry}(1) = LV_{exit}(1) \setminus \{x\}$
$LV_{entry}(2) = LV_{exit}(2) \setminus \{y\}$
$LV_{entry}(3) = LV_{exit}(3) \setminus \{x\}$
$LV_{entry}(4) = LV_{exit}(4) \cup \{x,y\}$
$LV_{entry}(5) = LV_{exit}(5) \setminus \{z\} \cup \{y\}$
$LV_{entry}(6) = LV_{exit}(6) \setminus \{z\} \cup \{y\}$
$LV_{entry}(7) = \{z\}$

$LV_{exit}(1) = LV_{entry}(2)$
$LV_{exit}(2) = LV_{entry}(3)$
$LV_{exit}(3) = LV_{entry}(4)$
$LV_{exit}(4) = LV_{entry}(5) \cup LV_{entry}(6)$
$LV_{exit}(5) = LV_{entry}(7)$
$LV_{exit}(6) = LV_{entry}(7)$
$LV_{exit}(7) = \emptyset$

ℓ	$LV_{entry}(\ell)$	$LV_{exit}(\ell)$
1	Ø	Ø
2	Ø	{y}
3	{y}	{x,y}
4	{x,y}	{y}
5	{y}	{z}
6	{y}	{z}
7	{z}	Ø

Analysis Result

May Analysis

Backward Analysis

Available Expressions

Data Flow Analysis

```
[x = a + b]1;
[y = a * b]2;
while ([y > a+b]3) {
      [a = a + 1]4;
      [x = a + b]5;
}
```

Input Program

ℓ	$kill_{AE}(\ell)$	$gen_{AE}(\ell)$
1	∅	{a+b}
2	∅	{a*b}
3	∅	{a+b}
4	{a+b, a*b, a+1}	∅
5	∅	{a+b}

$AE_{entry}(1) = \emptyset$
$AE_{entry}(2) = AE_{exit}(1)$
$AE_{entry}(3) = AE_{exit}(2) \cap AE_{exit}(5)$
$AE_{entry}(4) = AE_{exit}(3)$
$AE_{entry}(5) = AE_{exit}(4)$
$AE_{exit}(1) = AE_{entry}(1) \cup \{a+b\}$
$AE_{exit}(2) = AE_{entry}(2) \cup \{a*b\}$
$AE_{exit}(3) = AE_{entry}(3) \cup \{a+b\}$
$AE_{exit}(4) = AE_{entry}(4) \setminus \{ a+b, a*b, a+1\}$
$AE_{exit}(5) = AE_{entry}(5) \cup \{a+b\}$

ℓ	$AE_{entry}(\ell)$	$AE_{exit}(\ell)$
1	∅	{a+b}
2	{a+b}	{a+b, a*b}
3	{a+b}	{a+b}
4	{a+b}	∅
5	∅	{a+b}

Analysis Result

Must Analysis

Forward Analysis

Very Busy Expressions

Data Flow Analysis

```
if [a>b]1 {
    [x = b - a]2;
    [y = a - b]3;
} else {
    [y = b - a]4;
    [y = a - b]5;
}
```

Input Program

ℓ	$kill_{VB}(\ell)$	$gen_{VB}(\ell)$
1	∅	∅
2	∅	{b-a}
3	∅	{a-b}
4	∅	{b-a}
5	∅	{a-b}

$VB_{entry}(1) = VB_{exit}(1)$
$VB_{entry}(2) = VB_{exit}(2) \cup \{b-a\}$
$VB_{entry}(3) = \{a-b\}$
$VB_{entry}(4) = VB_{exit}(4) \cup \{b-a\}$
$VB_{entry}(5) = \{a-b\}$

$VB_{exit}(1) = VB_{entry}(2) \cap VB_{entry}(4)$
$VB_{exit}(2) = VB_{entry}(3)$
$VB_{exit}(3) = \emptyset$
$VB_{exit}(4) = VB_{entry}(5)$
$VB_{exit}(5) = \emptyset$

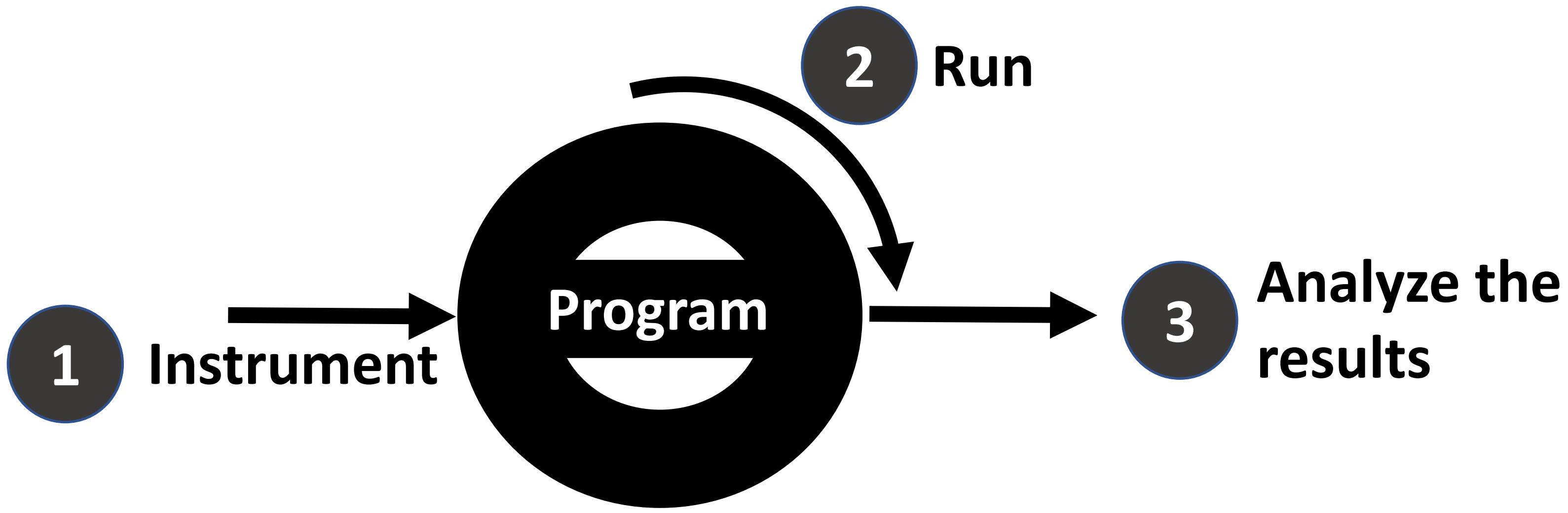
ℓ	$VB_{entry}(\ell)$	$VB_{exit}(\ell)$
1	{a – b, b – a}	{a – b, b – a}
2	{a – b, b – a}	{a – b}
3	{a – b}	∅
4	{a – b, b – a}	{a – b}
5	{a – b}	∅

Analysis Result

Must Analysis

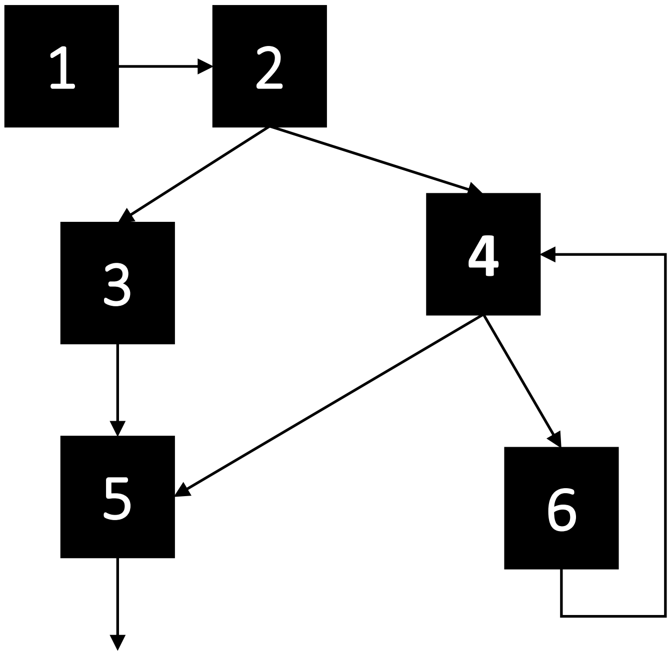
Backward Analysis

Dynamic Analysis



Example: Path Profiling

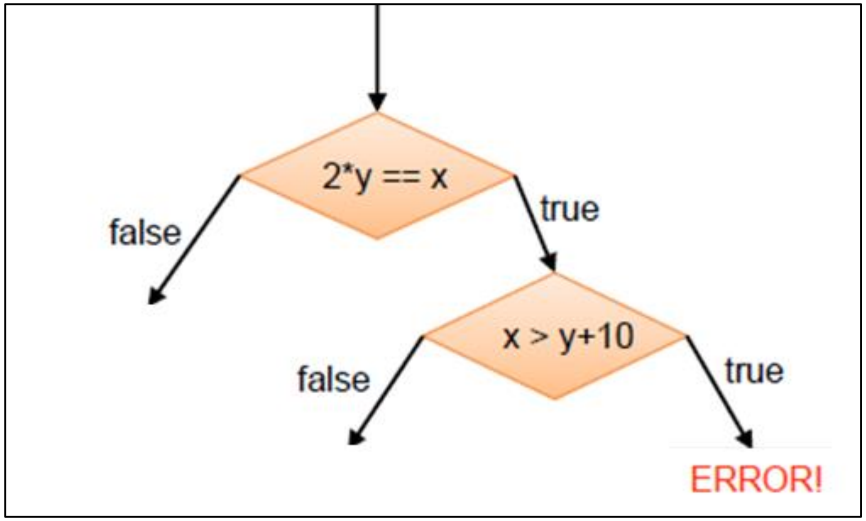
- Count the paths taken during actual execution
  - consider them for optimization, distribution (with better hardware support) and test coverage



Path	Frequency
1 2 3 5	100
1 2 4 6 4 5	2000
1 2 4 6 4 6 4 5	10
1 2 4 5	10

Another Example

```
z = 2*y ;
if (z == x)
{
    if (x > y+10)
        ERROR;
}
```



Execution Tree

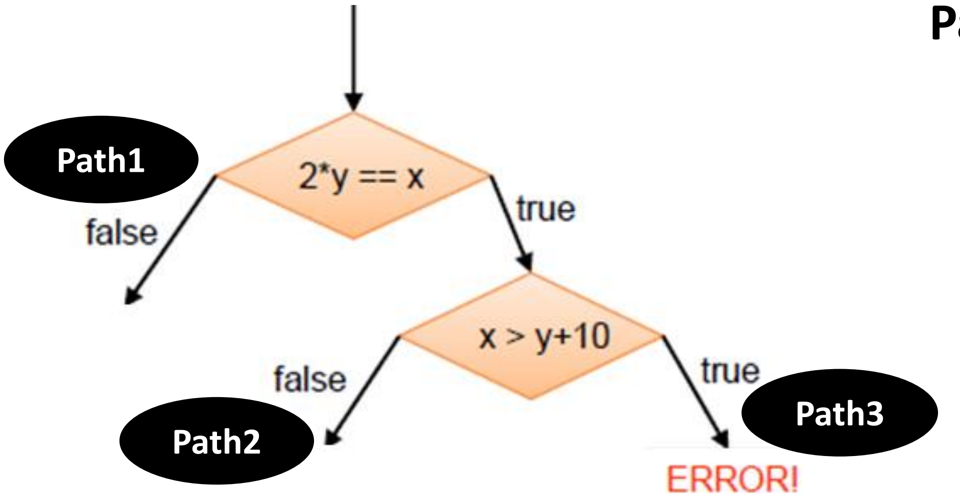
Symbolic variables

$$(x_0 = 2y_0) \wedge (x_0 > y_0 + 10)$$

A Path Constraint

Example taken from Symbolic Execution for Software Testing: Three Decades Later, Cadar and Sen.

Symbolic Execution



Path constraints for full path coverage

Path1: !C1,

Path2: C1 && C2,

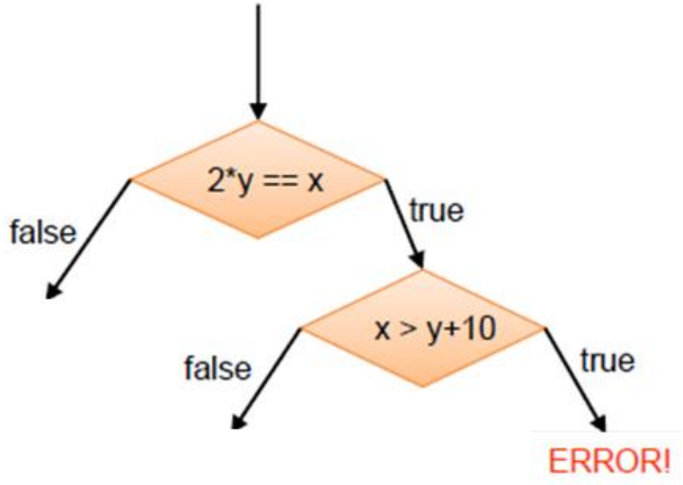
Path3: C1 && !C2

where

C1: x = 2y

C2: x > (y+10)

Symbolic Execution



Path Constraints	Constraint Solver Output
Path1: !C1	{x = 22, y = 7}
Path2: C1 && C2	{x = 2, y = 1}
Path3: C1 && !C2	{x = 30, y = 15}

Has applications in Automated Test Generation

Hybrid Analysis

- Often, we hit limitations with pure static or dynamic analysis.
- Hybrid = Static + Dynamic

Disadvantages of Symbolic Execution

- Constraints should be simple enough that a constraint solver is able to (efficiently) solve them.
  - E.g., (2^x) % large_prime == 13
- In the case of loops or recursion, we may need to put bounds on the number of iterations.
- Several custom functions may be uninterpreted.

Concolic Execution

- Concolic = Concrete + Symbolic
- Maintains both symbolic states as well as concrete states.

```
y= passwordHash(x);
if (z == x)
{
        ...
 } else {
        ERROR;
 }
```

Constraint solver cannot execute the uninterpreted function passwordHash(x)

Concolic Execution

```
hash = passHash(pass);
if  (x == hash)
{
        ...PATH1...
 } else {
        ...PATH2...
 }
```

Input	Concrete Execution
{pass = a, x = 1}	Execute passHash(“a”). Let it be 4000900977878888 Leads to PATH2.
{pass = a, x = 4000900977878888}	Leads to PATH1.

Concolic Execution

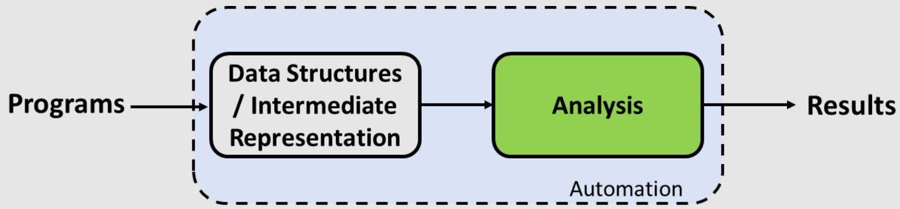
- Perform Symbolic Execution dynamically
- Run the program on concrete inputs.
  - One way is to start with random input values.
- Maintain a **concrete state** and a **symbolic state**

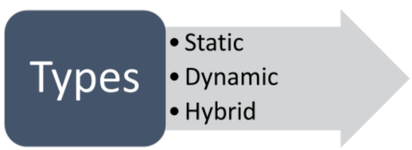
Hybrid Analysis

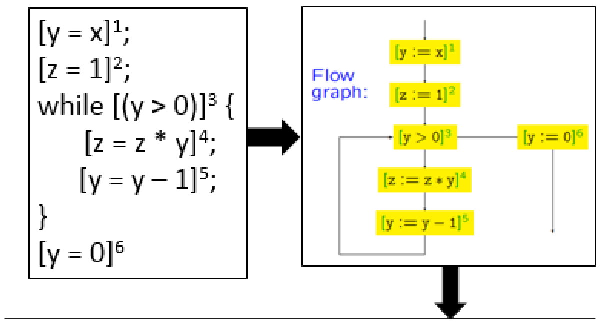
Research Trends

- Partial Program Analysis
- Scalable Program Analysis
- Language Independence

Summary

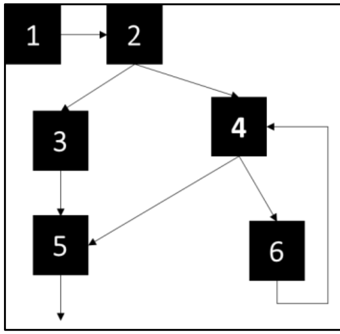






ℓ	$RD_{entry}(\ell)$	$RD_{exit}(\ell)$
1	(x,?),(y,?),(z,?)	(x,?),(y,1),(z,?)
2	(x,?),(y,1),(z,?)	(x,?),(y,1),(z,2)
3	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,2),(z,4),(y,5)
4	(x,?),(y,1),(z,2)(z,4),(y,5)	(x,?),(y,1),(z,4),(y,5)
5	(x,?),(y,1),(z,4),(y,5)	(x,?),(y,5),(z,4)
6	(x,?),(y,1),(z,2),(z,4),(y,5)	(x,?),(y,6),(z,2),(z,4)

Static Analysis



Path	Frequency
1 2 3 5	100
1 2 4 6 4 5	2000
1 2 4 6 4 6 4 5	10

Dynamic Analysis