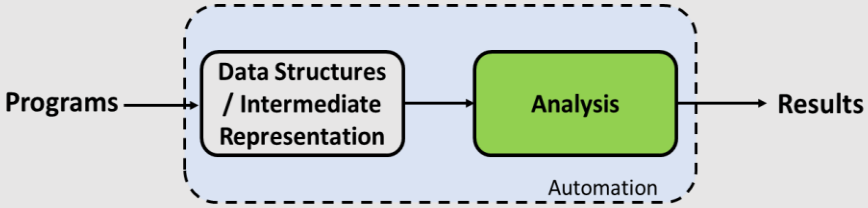
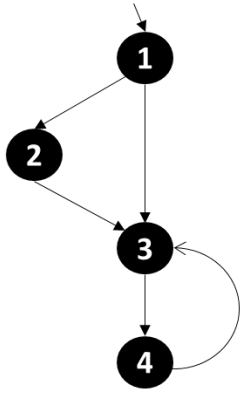
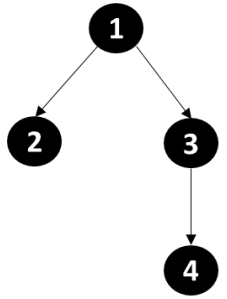




Program Analysis

Venkatesh Vinayakarao

venkateshv@cmi.ac.in
Mar – Apr, 2018
Chennai Mathematical Institute



"The theory of strings with concatenation has been widely argued as the basis of constraint solving for verifying string-manipulating programs. However, this theory is far from adequate for expressing many string constraints that are also needed in practice…" **– Chen et al., POPL 2018.**

Boeing 737 MAX Crash: Was Software To Blame And What Comes Next

Satellite data gathered from the Ethiopian Airlines flight and evidence from the crash site showed similarities with the Lion Air accident in Indonesia, which prompted the US Federal Aviation Administration to ground all Boeing MAX jets in service.

World | Reuters | Updated: March 15, 2019 10:59 IST


THE HINDU
MONDAY, MARCH 18, 2019

When software sends a plane into vertical dive

Boeing under attack for system flaw

JAGRITI CHANDRA

tion safety expert Captain Mohan Ranganathan

Quick Review – Program Representations

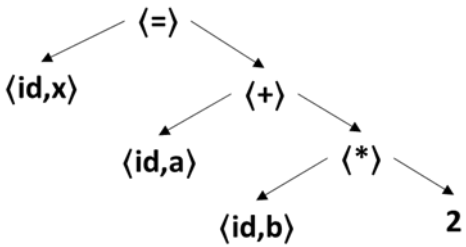
Lexemes

x = a + 2*b

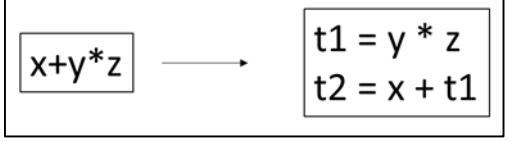
Tokens

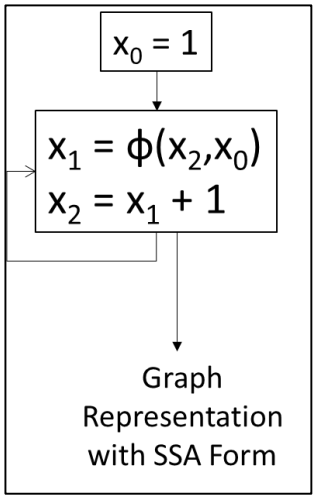
[(identifier, x), (operator, =), (identifier, a), (operator, +), (identifier, b), (operator, *), (literal, 2), (separator, ;)]

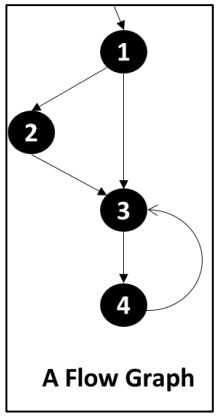
Abstract Syntax Tree

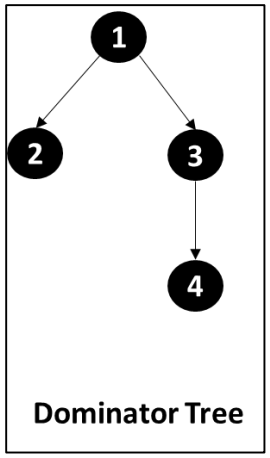


Three Address Code

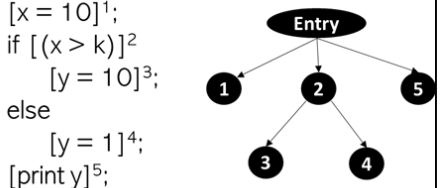


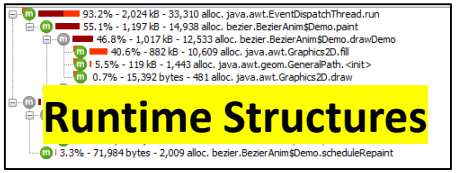






Control Dependence

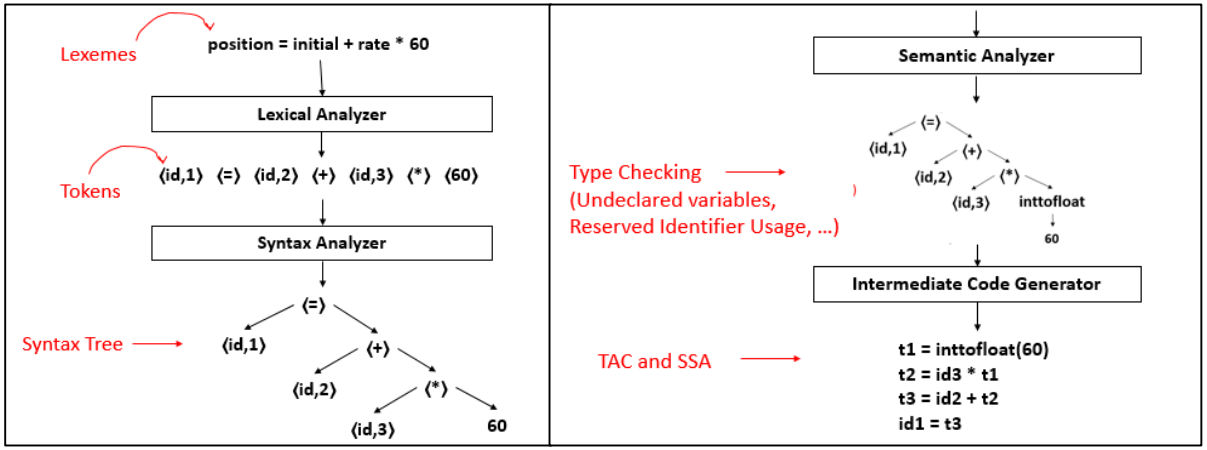




Tools

Eclipse JDT
Soot
JProfiler
Z3

The Optimization Phase



Parsed

No Compilation Errors!

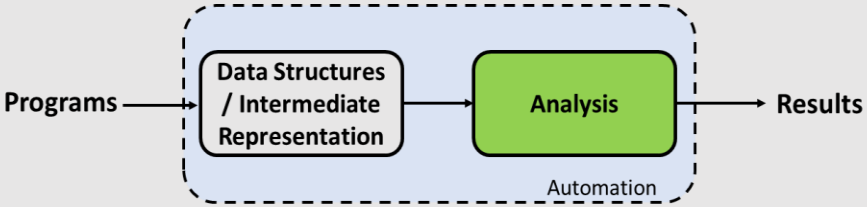
Intermediate Code Generation

Optimization

Code Generation

Compiled

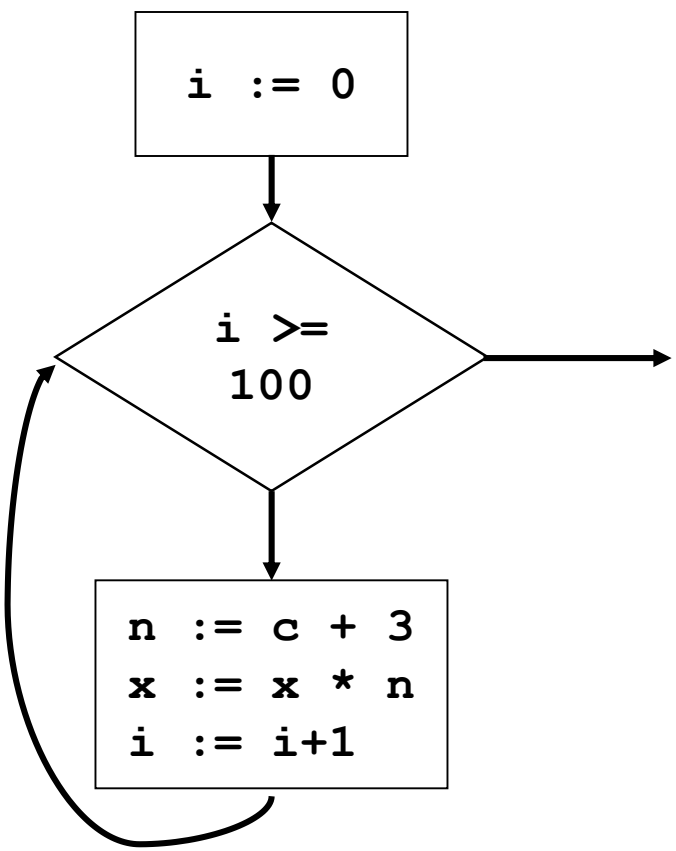
Now, we shall focus on optimization.



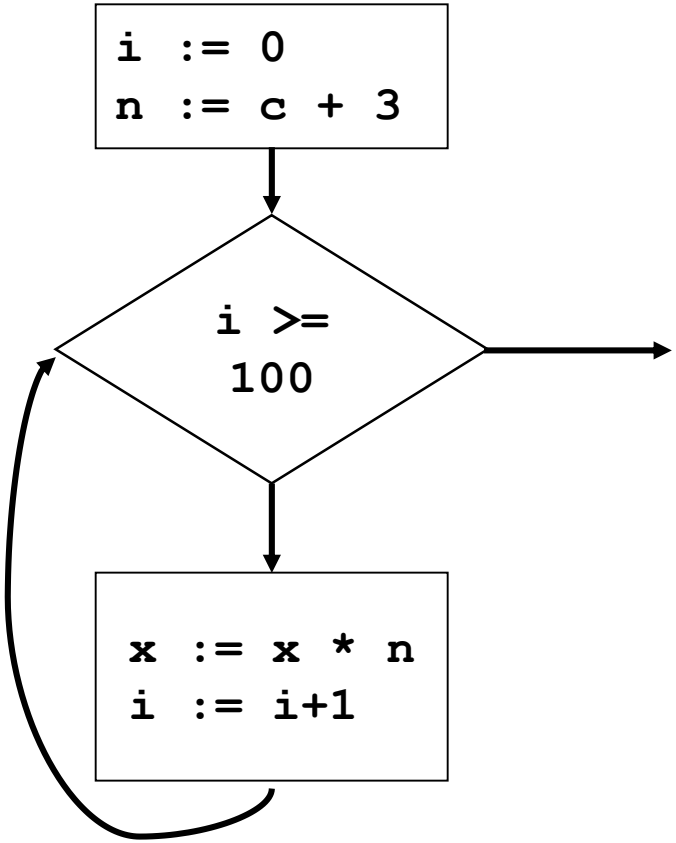
Static Analysis

Overview

Loop Invariant Code Motion



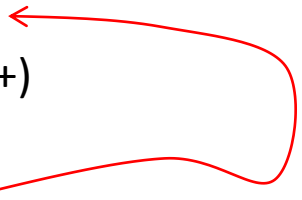
Before Optimization



After Optimization

Code Hoisting

```
void fun (int x, int y)
{
  for (int i = 0; i < 100; i++)
    {
      int temp = x + y;
      System.out.println(temp);
    }
}
```

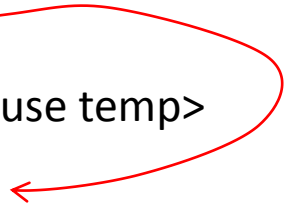


This line can be moved upwards
(Code Hoisting)

Can you come up with an example for code sinking?
We need to move the code after the loop.

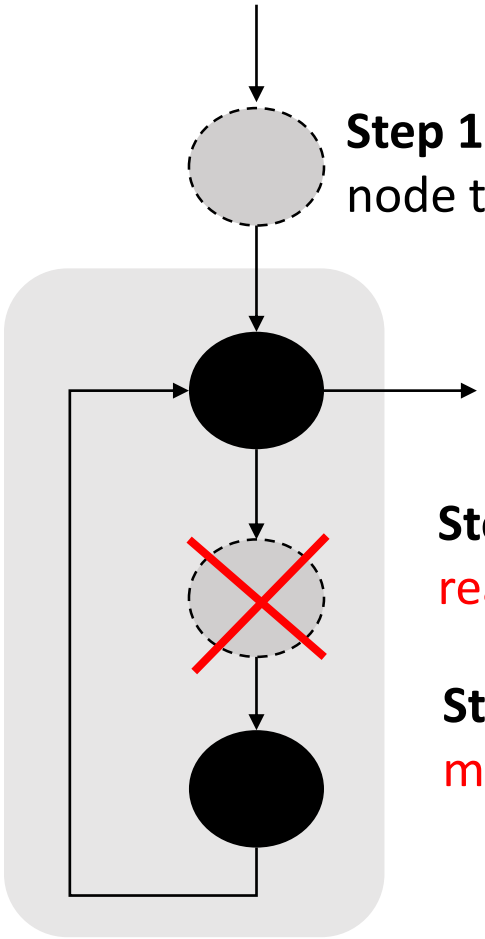
Code Sinking

```
void fun (int x, int y)
{
  for (int i = 0; i < 100; i++)
  {
      int temp = x + y;
      ...<more lines that do not use temp>
  }
   System.out.println(temp);
}
```



This line can be moved downwards
(Code Sinking)

A Code Hoisting Algorithm

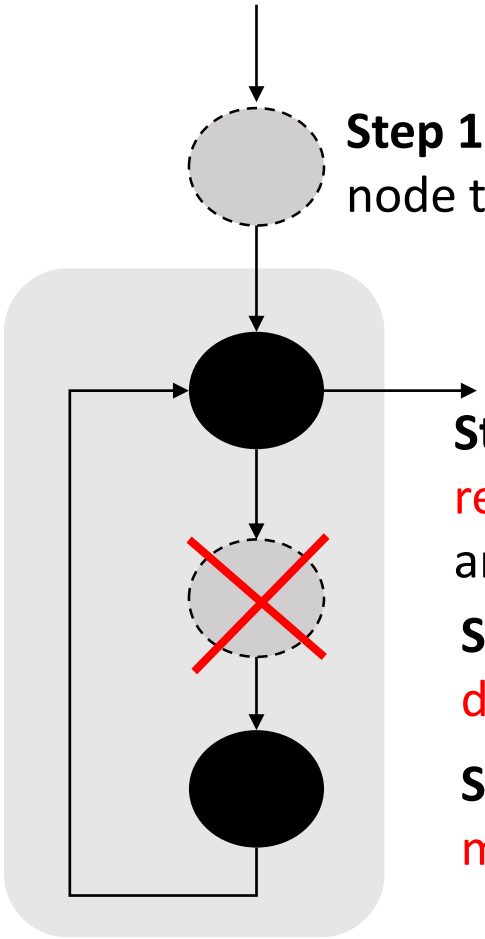


Step 1: Insert a pre-header node to hoist code.

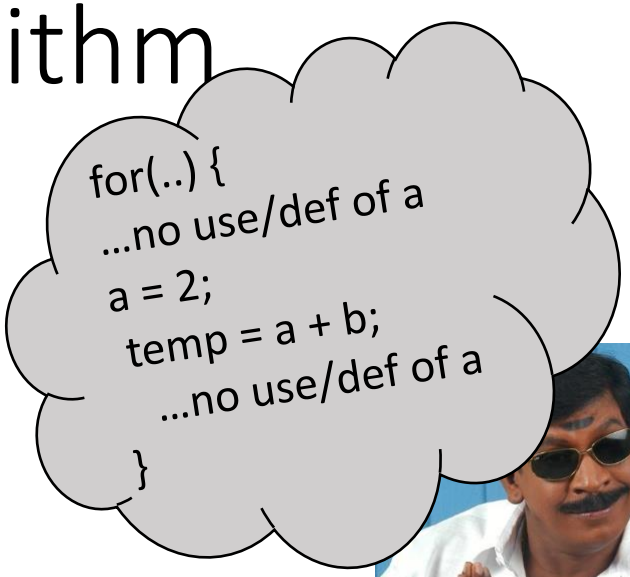
Step 2: Find expressions whose operands have reaching definitions from outside the loop.

Step 3: Remove such statements and move them to the pre-header node.

A Code Hoisting Algorithm



Step 1: Insert a pre-header node to hoist code.



Step 2: Find expressions whose operands have reaching definitions from outside the loop, or are constants. Mark them as loop invariant.

Step 2.5: Operands have exactly one reaching definition and that definition is loop invariant.

Step 3: Remove such statements (S) and move them to the pre-header node.

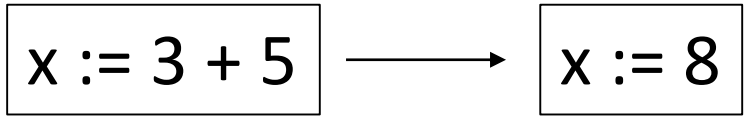
Beware of Function Calls

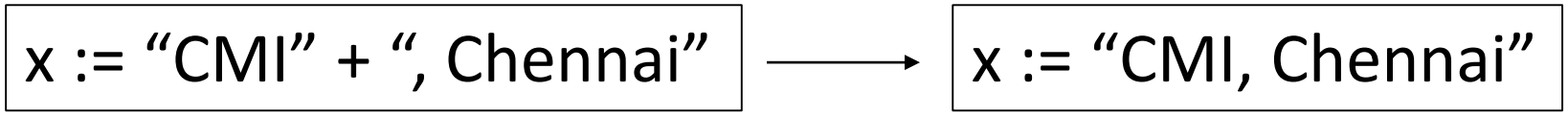
```
void fun (int x, int y)
{
    ...
    for (int i=0; i<=10; i++) {
        ...
        int value = rand.nextInt(50);
        ...
    }
    ...
}
```

Same method.
Same parameters.
Different results.
So, we cannot move such lines.

Compilers Know Math!

Constant Folding

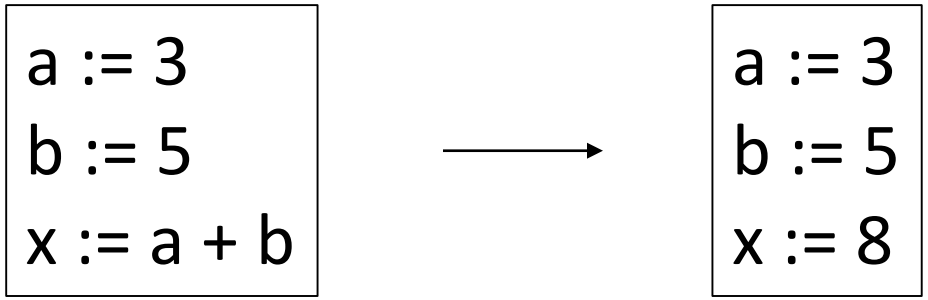


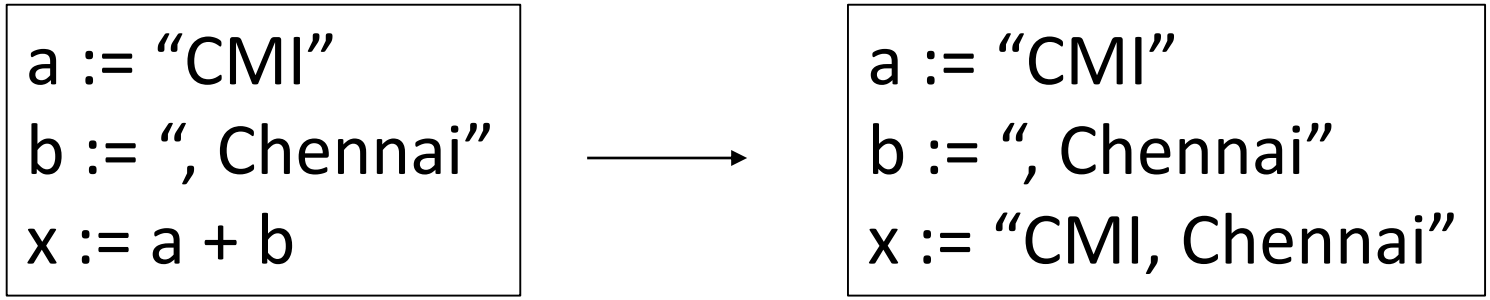


They can do string operations too!

Constant Propagation

Dead Code Elimination can remove a and b.





Substituting the constant value to a variable

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;
```

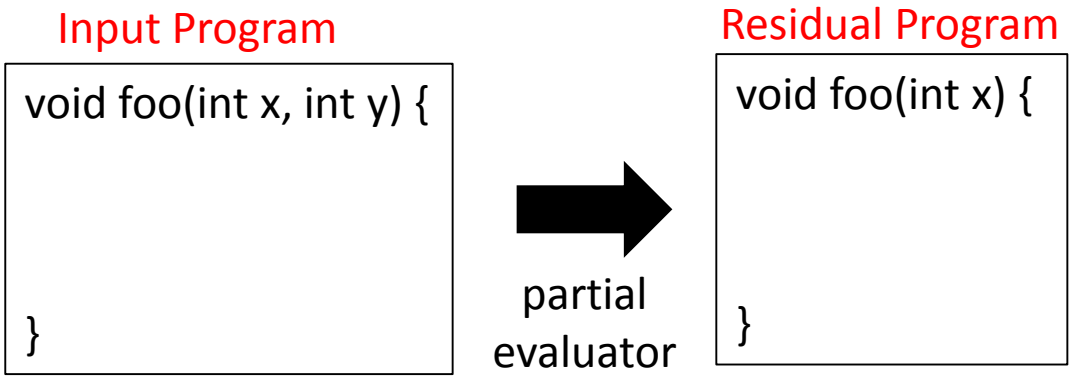
→

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

```
return 4;
```

Partial Evaluation

Let us say, y is known to be fixed (a.k.a., configured) every time we execute this program.



$prog: I_{fixed} \times I_{dynamic} \rightarrow O$ $prog^{*}: I_{dynamic} \rightarrow O$

where I and O are sets of inputs and Outputs respectively.

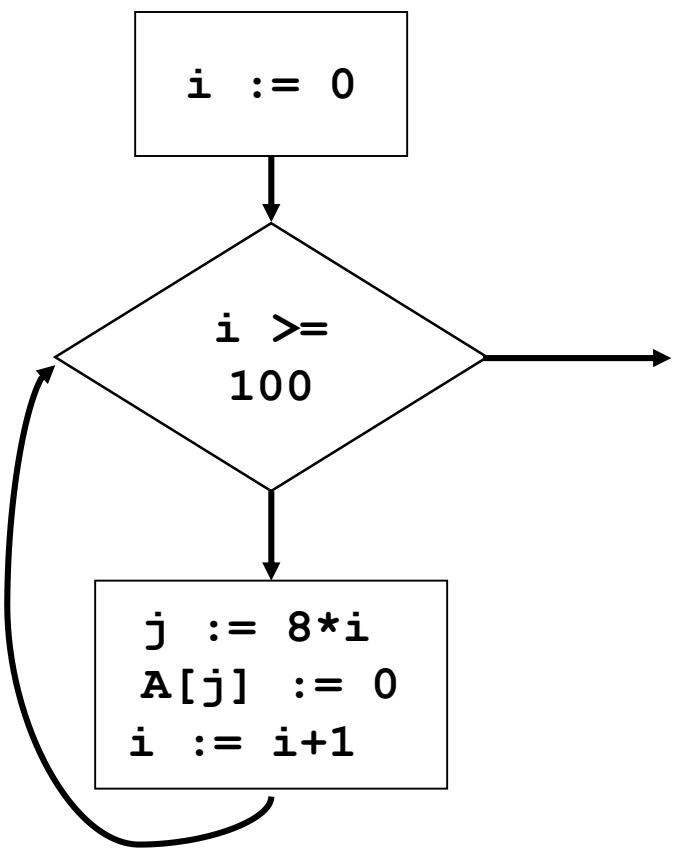
Partial Evaluation and Automatic Program Generation **(Book is freely available online)**

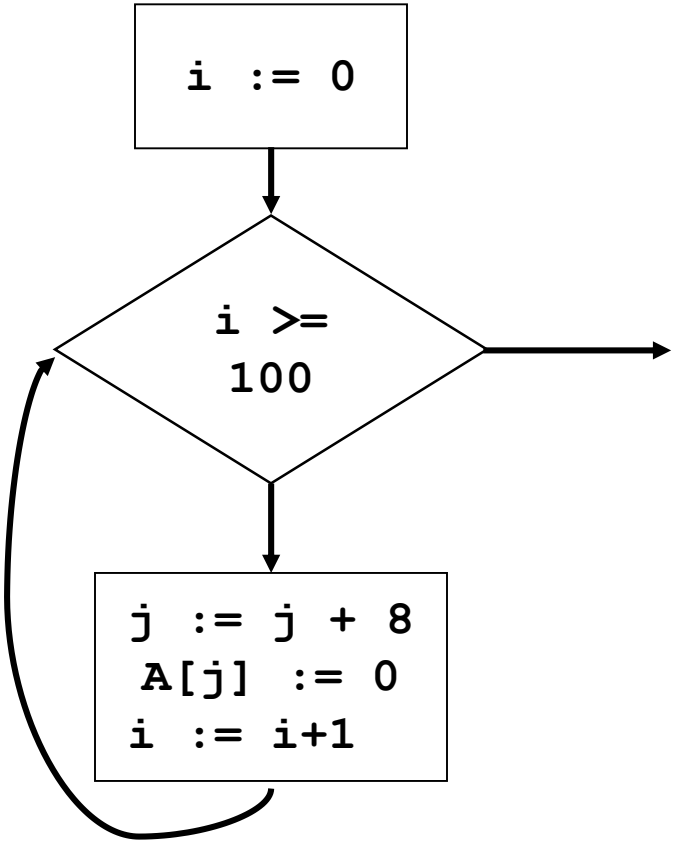
Kleene's Smn Theorem

Cross Product

- What is the cross product of two sets, A = {a,b,c} and Z = {1,2,3}?
  - A x Z = ?

Strength Reduction





+ is less expensive than *

Graded Quiz

- Optimize the following snippet

```
void foo(int x, int y) {
      y = x * 15;
      System.out.println(y);
}
```

Peephole Optimization

Before	After
y = x / 8	y = x >> 3
y = x * 64	y = x << 6
y = x * 2	y = x << 1

McKeeman, Peephole Optimization, 1965.

More Examples

- Can you find the improvement?

```
int factorial(int i)
 {
    if (i == 0)
       return 1;
    else
       return i *
          factorial(i - 1);
 }
```

```
static int factorialTable[] =
     {1, 1, 2, 6, 24, 120, 720 /*
etc */};

int factorial(int i)
{
     return factorialTable[i];
}
```

PhD Thesis, Venkatesh Vinayakarao.

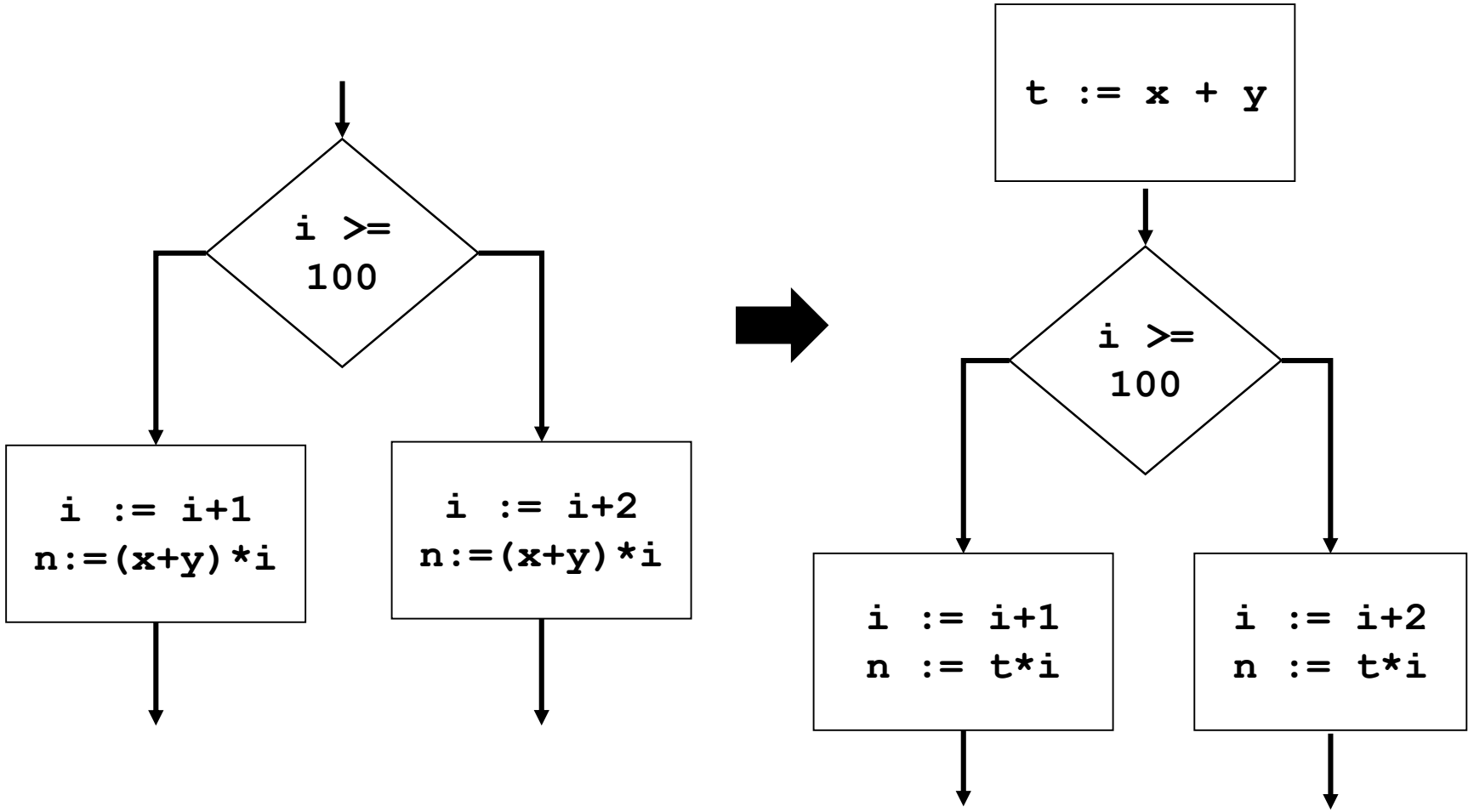
More Examples

- **Can you find a variant?**

```
int foo(int n, int r) {
  if (n > 0) {
        return (n%r + foo (n/r, r ));
  }
  else return 0;
}

void main() {
   printf("%d ", foo(346,10));
}
```

Common Subexpressions



More Examples

- What could be improved here?

```
float x,y,a,b,c,d;
…
x = (a/b)*c;
y = (a/b)*d;
```

- Common Sub-Expression (a/b) can be computed only once.

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

Beyond Compiler Optimizations

Applications

- Apart from bug detection and optimization, static program analysis techniques are used in
  - Source Code Plagiarism Detection (Moss, …)
  - Clone Detection
  - Code Search
  - Automated MOOC Assignment Feedback
  - Automated Programming Quiz
  - Program Translators
  - Code Completion
  - Refactoring
  - and so on…

Readings

- **Before the next lecture, please read**
  - Chapters 1 and 2 of Anders Møller and Michael I. Schwartzbach‘s book on Static Program Analysis.

Data Flow Analysis

The Classic Four!

WHILE Language

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

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

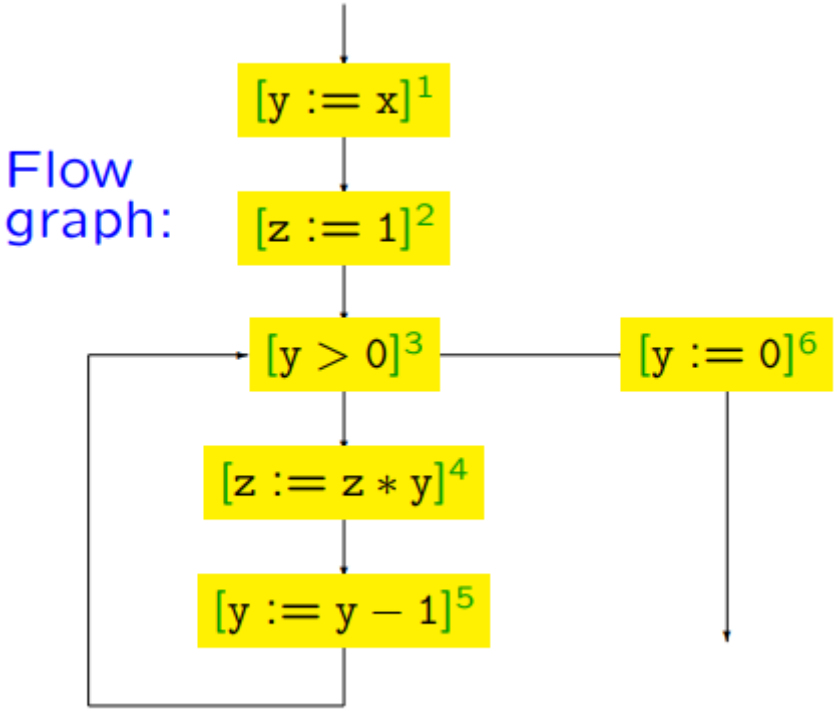
$$b ::= true \mid false \mid not\ b \mid b_1\ op_b\ b_2 \mid a_1\ op_r\ a_2$$

$$S ::= [x := a]^l \mid [skip]^l \mid S_1; S_2 \mid$$
$$\text{if } [b]^l \text{ then } S_1 \text{ else } S_2 \mid \text{while } [b]^l \text{ do } S$$

Labeled Programs and Control Flow

$[y := x]^1$;
$[z := 1]^2$;
while $[y > 0]^3$ **do**
    $[z := z * y]^4$;
    $[y := y - 1]^5$
od;
$[y := 0]^6$

Flow graph:



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

Reaching Definitions

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?

Notations

$[y := x]^1;$
$[z := 1]^2;$
while $[y > 0]^3$ **do**
    $[z := z * y]^4;$
    $[y := y - 1]^5$
od;
$[y := 0]^6$

$[y = x]^1;$
$[z = 1]^2;$
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		
3		
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		
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		
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)

Labeled
Input Program

$[y = x]^1$;
$[z = 1]^2$;
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 $\mathbf{Lab} = \{1,2,3,4,5,6\}$

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, $\overrightarrow{RD}$ = $RD_{entry}(1),\ldots,RD_{exit}(6)$
- $\overrightarrow{RD}$ = F(RD)

A Simple Iterative Algorithm

$\overrightarrow{RD}$ = (∅,…. ∅)

j = 0;

while $\overrightarrow{RD}$ ≠ F(RD_1, · · · , RD_{12})

do $\overrightarrow{RD}$:= F(RD_1, · · · , 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	∅	∅

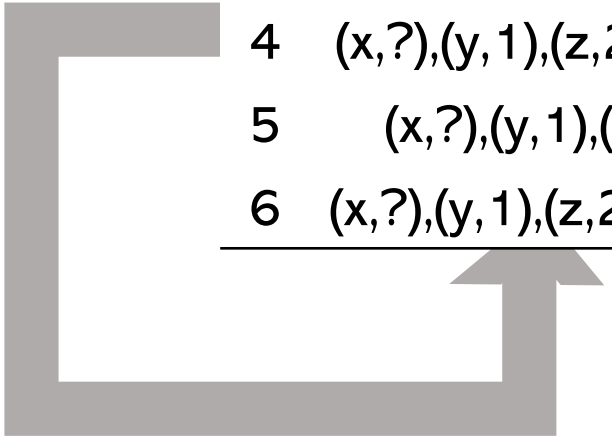
F(RD)

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

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

Reaches a Least Fixed Point

F(RD)



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)

The Question

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

$$
\begin{array}{l}
[\mathtt{y} := \mathtt{x}]^{1};\\
[\mathtt{z} := 1]^{2};\\
\mathtt{while}\ [\mathtt{y} > 0]^{3}\ \mathtt{do}\\
\quad [\mathtt{z} := \mathtt{z} * \mathtt{y}]^{4};\\
\quad [\mathtt{y} := \mathtt{y} - 1]^{5}\\
\mathtt{od};\\
[\mathtt{y} := 0]^{6}
\end{array}
$$

Answer: No!

Since, $RD_{entry}(5)$ = (x,?),(y,1),(z,4),(y,5)

There is no (z,2) in it.