Week 3 Static Analysis Examples

0. Constant Propagation (demo)

- Flat lattice of integer values
- Rules are same as Sign analysis, but with wider lattice.
- Note: this is a **Forward analysis**

	Iteration 1	Iteration 2	Iteration 3
var x,y,z;	[x->T, y->T, z->T]	same	
x = 27;	[**x -> 27**, y->T, z->T]	same	
y = input;	[x->27, **y->T**, z->T]	...	
z = 2*x+y;	[x->27, y->T, z->T]		
~~if (x < 0)~~	~~eval(x<0) = [x->27, y->T, z->T] = {true, false}~~		
~~{ y = z-3; }~~	~~[x->27, y->T, z->T] (A)~~		
~~else~~			
{ y = 12; }	[x->27, y->12, z->T] (B)		
;	~~join(A,B) = [x->27, y->T, z->T]~~		
output y;			

1. Live variables analysis

- A variable is *live* at a program point if its current value may be read in the remaining execution.
- *(see Slide 7 in 4-flow-sensitive-analysis.pdf)*
- **Note: Backward May analysis! JOIN = union.** $L = (2^{\{x,y,z\}}, \subseteq)$

	Iteration 1	Iteration 2 (the fix point)
var x,y,z;	{}\{x,y,z}={}	
x = input;	{x}\{x}u{} = {}	
while (x>1) {	{x}	JOIN({x},{x})={x}
y = x/2;	{x}	
if (y>3)	{x,y}	
x = x-y;	{x,y}	
z = x-4;	{x,z}\{z}u{x} = {x}	
if (z>0)	{x,z}u{z} = {x,z}	
x = x/2;	{x,z}\{x}u{x} = {x,z}	
~~z = z-1;~~	{x}\{z}u{z} = {x,z}	
}		
output x;	{x}	{x}
exit	{}	{}

2. Available expressions analysis

- a nontrivial expression is *available* at a program point if its current value must have been computed earlier in the execution
- *(see Slide 18 in 4-flow-sensitive-analysis.pdf)*
- **Note**: **FORWARDS** analysis! JOIN=intersection (a **MUST** analysis)
- Note: when calculating fix-points, start any unknown nodes as ⊥. That is {a+b,y>a,a+1,a*b}

$L=2^{\{a+b,y>a,a+1,a^*b\}}, \supseteq)$

	Iteration 1	Iteration 2 (the fix point)
`var x,y,z,a,b;`	{}	{}
`z = a+b;`	{a+b}	{a+b} // (so we can reuse this for y)
`y = a+b;`	{a+b}	{a+b}
`while (y > a) {`	JOIN({a+b},{}) ∪ {y>a} = {a+b,y>a}	JOIN({a+b},{a*b}) ∪ {y>a} = {y>a}
`a = a+1;`	{a+b,y>a,a+1}|a = {}	{y>a,a+1}|a = {}
`x = a*b;`	{a*b}↓x = {a*b}	{a*b}↓x = {a*b}
`}`		
`<exit>`	{y>a}	{y>a}

3. Very busy expressions analysis

- a nontrivial expression is *very busy* if it will definitely be evaluated before its value changes
- *(see Slide 28 in 4-flow-sensitive-analysis.pdf)*
- **Note**: **BACKWARDS** analysis! JOIN=intersection (i.e. a **MUST** analysis)
- (we use same lattice as previous analysis, but based on all expressions in this program)

$L=2^{\{x-1,x-2,x>0,a*b,a*b-x\}}, \supseteq)$

```
var x,a,b;              {
x = input;              {
a = x-1;                {
b = x-2;                {
while (x > 0) {         {
 output a*b-x;      {
 x = x-1;               {
}
output a*b;             {
```

4. Reaching definitions analysis (def-use)

- The *reaching definitions* for a program point are those assignments that may define the current values of variables.
- *(see Slide 35 in 4-flow-sensitive-analysis.pdf)*
- **Note**: **FORWARDS** analysis! JOIN=union (i.e. a **MAY** analysis)

The powerset lattice of assignments

$L = (2^{\{x=input,\ y=x/2,\ x=x-y,\ z=x-4,\ x=x/2,\ z=z-1\}}, \subseteq)$

```
var x,y,z;          {
x = input;          {
while (x > 1) {
 y = x/2;           {
 if (y>3)
  x = x-y;      {
 z = x-4;           {
 if (z>0)
  x = x/2;      {
 z = z-1;           {
}
output x;           {
```