Taller 2 - Implementando An´alisis de Dataflow

Grupo #6

Nombre	Mail	LU
Manuel Panichelli	panicmanu@gmail.com	72/18
Elias Cerdeira	eliascerdeira@gmail.com	692/12

Parte 1 - Definiendo el Zero Analysis

Ejercicio 1 - Asignación de constante

n	OUT[n](x)
x = 0	<x, ZERO>
x = k	<x, NOT_ZERO>

Donde $K \in Z - \{0\}$

Ejercicio 2 - Asignación de variable

IN[n](y)	OUT[n](x)
$\bot$	$\bot$
Z	Z
NZ	NZ
MZ	MZ

Ejercicio 3 - Suma

IN[n](y)	IN[n](z)	OUT[n](x)
$\bot$	$\bot$	$\bot$
Z	$\bot$	$\bot$
NZ	$\bot$	$\bot$
MZ	$\bot$	$\bot$
$\bot$	Z	$\bot$
Z	Z	Z
NZ	Z	NZ
MZ	Z	MZ

IN[n](y)	IN[n](z)	OUT[n](x)
⊥	NZ	⊥
Z	NZ	NZ
NZ	NZ	MZ
MZ	NZ	MZ
⊥	MZ	⊥
Z	MZ	MZ
NZ	MZ	MZ
MZ	MZ	MZ

Ejercicio 4 - Resta

IN[n](y)	IN[n](z)	OUT[n](x)
⊥	⊥	⊥
Z	⊥	⊥
NZ	⊥	⊥
MZ	⊥	⊥
⊥	Z	⊥
Z	Z	Z
NZ	Z	NZ
MZ	Z	MZ
⊥	NZ	⊥
Z	NZ	NZ
NZ	NZ	MZ
MZ	NZ	MZ
⊥	MZ	⊥
Z	MZ	MZ
NZ	MZ	MZ
MZ	MZ	MZ

Ejercicio 5 - Multiplicación

IN[n](y)	IN[n](z)	OUT[n](x)

IN[n](y)	IN[n](z)	OUT[n](x)
⊥	⊥	⊥
Z	⊥	⊥
NZ	⊥	⊥
MZ	⊥	⊥
⊥	Z	⊥
Z	Z	Z
NZ	Z	Z
MZ	Z	Z
⊥	NZ	⊥
Z	NZ	Z
NZ	NZ	NZ
MZ	NZ	MZ
⊥	MZ	⊥
Z	MZ	Z
NZ	MZ	MZ
MZ	MZ	MZ

Ejercicio 6 - División

IN[n](y)	IN[n](z)	OUT[n](x)
⊥	⊥	⊥
Z	⊥	⊥
NZ	⊥	⊥
MZ	⊥	⊥
⊥	Z	⊥
Z	Z	⊥
NZ	Z	⊥
MZ	Z	⊥
⊥	NZ	⊥
Z	NZ	Z
NZ	NZ	NZ

IN[n](y)	IN[n](z)	OUT[n](x)
MZ	NZ	MZ
⊥	MZ	⊥
Z	MZ	Z
NZ	MZ	NZ
MZ	MZ	MZ

Nota al pie: deseamos ser optimistas y propagar el valor que se obtendría en caso de que MZ no haya sido Z.

Ejercicio 7

```
public int productoria(int y) {
    int x = y;
    y = 1;

    while (x != 1) {
        y = x * y;
        x = x - 2;
    }

    return y;
}
```

n	IN[n](x)	IN[n](y)	OUT[n](x)	OUT[n](y)
1	⊥	MZ	⊥	MZ
2	⊥	MZ	MZ	MZ
3	MZ	MZ	MZ	NZ
4	MZ	MZ	MZ	MZ
5	MZ	MZ	MZ	MZ
6	MZ	MZ	MZ	MZ
7	MZ	NZ	-	-

Parte 2 - Implementando el Zero Analysis en SOOT

Merge (supremo)

⊔	⊥	Z	NZ	MZ
⊥	⊥	Z	NZ	MZ

⊔	⊥	Z	NZ	MZ
Z	Z	Z	MZ	MZ
NZ	NZ	MZ	NZ	MZ
MZ	MZ	MZ	MZ	MZ

Union (infimo)

⊓	⊥	Z	NZ	MZ
⊥	⊥	⊥	⊥	⊥
Z	⊥	Z	⊥	Z
NZ	⊥	⊥	NZ	NZ
MZ	⊥	Z	NZ	MZ

Para compilar o buildear el proyecto del análisis es necesario ejecutar el siguiente comando.

```
mvn install
```

Una vez realizado esto, procedemos a ejecutar el análisis sobre cada uno de los ejemplos del enunciado utilizando los siguientes comandos.

```
java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest1 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest1 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest2 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest3 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest4 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest5 -v -print-
tags -p jtp.DivisionByZeroAnalysis on
```

```
java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest6 -v -print-
tags -p jtp.DivisionByZeroAnalysis on

java -jar zero-analysis/target/zero-analysis-1.0-SNAPSHOT-jar-with-
dependencies.jar -cp .:./examples/:$JRE -f J ZeroAnalysisTest7 -v -print-
tags -p jtp.DivisionByZeroAnalysis on
```

La salida obtenida en cada caso puede encontrase en el directorio `soot-dataflow-analysis/sootOutput`.

Para cada caso mostramos el código del ejemplo y el conjunto de entrada para la instrucción `return`.

Ejemplo 1

```
public class ZeroAnalysisTest1 {
    public static int test1(int m, int n) {
        int x = 0;
        int k = x * n;
        int j = m / k; /*Possible division by zero here*/

        return j;
    }
}
```

No aparece `x` en el output de `Soot` ya que el compilador reemplaza su uso por la constante `0`.

```
/*ZeroAbstractSet{{k=zero, m=maybe-zero, j=bottom, n=maybe-zero}}*/
```

Ejemplo 2

Observamos que en este caso la instrucción `int i = x + m` no aparece en la salida de la corrida de `Soot`. Creemos que esto se debe a que se inicializa una variable que nunca se utiliza, por lo que el compilador optimiza el código removiendo la instrucción.

```
public class ZeroAnalysisTest2 {
    public static int test2(int m, int n) {
        int x = n - n;
        int i = x + m;
        int j = m / x; /*Possible division by zero here*/

        return j;
    }
}
```

```
/*ZeroAbstractSet{{x=maybe-zero, m=maybe-zero, n=maybe-zero, j=maybe-
zero}}*/
```

Ejemplo 3

```
public class ZeroAnalysisTest3 {
    public static int test3(int m, int n) {
        int x = 0;
        int j = m / n; /*Possible division by zero here*/
        return j;
    }
}
```

El compilador remueve la instrucción `int x = 0;` ya que no tiene ningún efecto. Por lo tanto, no hay información de *data flow* para esa variable.

```
/*ZeroAbstractSet{{j=maybe-zero, m=maybe-zero, n=maybe-zero}}*/
```

Ejemplo 4

```
public class ZeroAnalysisTest4 {
    public static int test4(int m, int n) {
        int x = 0;
        if (m != 0){
            x = m;
        } else {
            x = 1;
        }
        int j = n / x; /*Possible division by zero here*/
        return j;
    }
}
```

```
/*ZeroAbstractSet{{x=maybe-zero, m=maybe-zero, j=maybe-zero, n=maybe-
zero}}*/
```

Ejemplo 5

```
public class ZeroAnalysisTest5 {
    public static int test5(int y) {
        int x = y;
```

```
        y = 1;
        while (x != 1) {
            y = x*y;
            x = x-1;
        }
        return y;
    }
}
```

En este ejemplo notamos que para poder realizar las operaciones de asignación que en el cuerpo usan la misma variable a la que asignan el valor se utilizan variables temporales para poder descomponer en suboperaciones. Por ejemplo

```
int y = x * y

# Lo descompone en

temp$1 = x * y
y = temp$1
```

```
/*ZeroAbstractSet{{x=maybe-zero, y=maybe-zero}}*/
```

Ejemplo 6

```
public class ZeroAnalysisTest6{
    public static int test6(int x) {
        int y;
        if (x == 0) {
            y = 1;
        } else {
            y = 2;
        }
        int r = x/y;
        return r;
    }
}
```

```
/*ZeroAbstractSet{{x=maybe-zero, y=not-zero, r=maybe-zero}}*/
```

Ejemplo 7

```
public class ZeroAnalysisTest7 {
    public static int test7() {
        int i = 0;
        int j = 1;
        int d = j/i; /*Possible division by zero here*/

        if (d > 0) {
            d = 1;
        }

        return d;
    }
}
```

Observamos que con respecto al output esperado no aparecen las variables `i` y `j`, ya que el compilador las reemplaza por las constantes 0 y 1 respectivamente.

```
/*ZeroAbstractSet{{d=not-zero}}*/
```