Ingenieria del Software II

Segundo Cuatrimestre de 2020

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Taller 1

SOOT

Integrantes	LU	Correo electrónico
Tripodi, Guido	843/10	guido.tripodi@hotmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Ejercicio 1

En base a la definición del análisis **Reaching Definition**, el conjunto IN y OUT para cada nodo del Grafo de Control de Flujo se define:

$$IN[n] = \bigcup OUT[n'] \tag{1}$$

Donde n' son los predecesores de n

$$OUT[n] = (IN[n] - KILL[n]) \cup GEN[n] \tag{2}$$

1.1. Control-Flow Graphs

En base al código de este ejercicio se realizo el gráfico de control de flujo en cuestión, el cual se muestra a continuación.

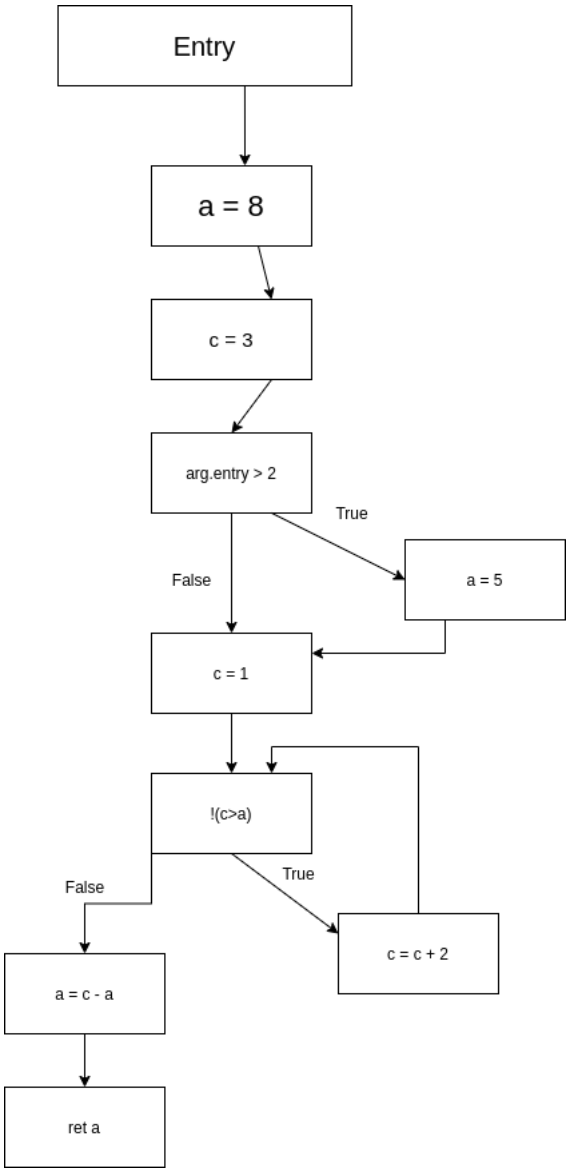

Figura 1: Control-Flow Graphs

1.2. ¿Que definiciones de las variables a y c alcanzan la lınea #5?

Realizando un análisis de Data-Flow llegamos a los siguientes resultados:

IN	OUT
<a,1>,<c,5>,<c,7>,<a,3>	<c,7>,<c,5>,<a,9>

Teniendo en cuenta el código notamos que la variable a toma un valor en la linea 1 y puede tomar otro valor en la linea 3 el cual depende si se cumple la guarda del if de la linea 2. Luego, como este tipo de análisis es *forward*, el out de la linea 5 "pisara.[el] valor de c de la linea 1 por el valor de la linea 5.

Luego, si tenemos en cuenta los resultados obtenidos por la herramienta *SOOT* podemos notar que en dicha linea 5 obtiene resultados distintos ya que toma un código optimizado el cual elimina lineas que no ejecuta.

1.3. ¿Que definiciones de las variables a y c alcanzan la lınea #9?

Realizando un análisis de Data-Flow llegamos a los siguientes resultados:

IN	OUT
<a,1>,<a,3>,<c,5>,<c,7>	<a,9>,<c,5>,<c,7>

Teniendo en cuenta el código notamos que la variable a toma un valor en la linea 1 y puede tomar otro valor en la linea 3 el cual depende si se cumple la guarda del if de la linea 2. Con el mismo razonamiento, la variable c toma un valor en la linea 1, el cual es "pisado.[en] la linea 5, y puede tomar un nuevo valor en la linea 7 si es que cumple la condición del while. Si tenemos en cuenta el out de la linea cosultada, notamos que se modifica el valor de a, por lo cual pisa los anteriores.

Luego, al igual que en el item anterior, realizando una comparación con los resultados obtenidos por la herramienta *SOOT* podemos notar que en dicha linea 9 obtiene resultados distintos teniendo en cuenta que utiliza un código optimizado pero de igual forma realiza un análisis reaching def, en el cual analiza los posibles resultados de la variable c y a y devuelve que los valores posibles de estas variables son 1 y c + 2 para c y 8 y 5 para a, estos valores son obtenidos a través de las lineas 1 3 5, y 7, notando también que se elimina el valor de c igual a 3 que se obtiene en la linea 1.

1.4. ¿Que definiciones de las variables a y c alcanzan la lınea #10?

Realizando un análisis de Data-Flow llegamos a los siguientes resultados:

IN	OUT
<a,9>,<c,5>,<c,7>	<a,9>

Continuando con el análisis del código notamos que la variable a toma un nuevo valor en la linea 9 pisando los anteriores mientras que la variable c se queda con los valores de la linea 5 y

potencialmente de la linea 7 (dependiendo de si puede o no ingresar al ciclo). Luego, como el código retorna una variable, el out de esta linea no sera vacío sino que tendrá el valor final de la linea 9 para la variable a.

Ahora, si tenemos en cuenta los resultados obtenidos por la herramienta SOOT y la optimización que realiza en las lineas de código (la linea 10 de código no es la misma que la linea 10 de código optimizada), la linea equivalente en cuestión para el output de SOOT tendrá como in a las variables a y c con los valores obtenidos en las lineas 1 3 5, y 7 y como out devuelve la variable a con el valor final obtenido en la linea 9 pisando los anteriores.

2. Ejercicio 2

Para este ejercicio trabajamos utilizando el análisis de tipo **Live Variables**. En dicho análisis el conjunto IN y OUT para cada nodo del Grafo de Control de Flujo se define de la siguiente manera:

$$OUT[n] = \bigcup IN[n'] \tag{3}$$

Donde n' son los sucesores de n

$$IN[n] = (OUT[n] - KILL[n]) \cup GEN[n] \tag{4}$$

2.1. Control-Flow Graphs

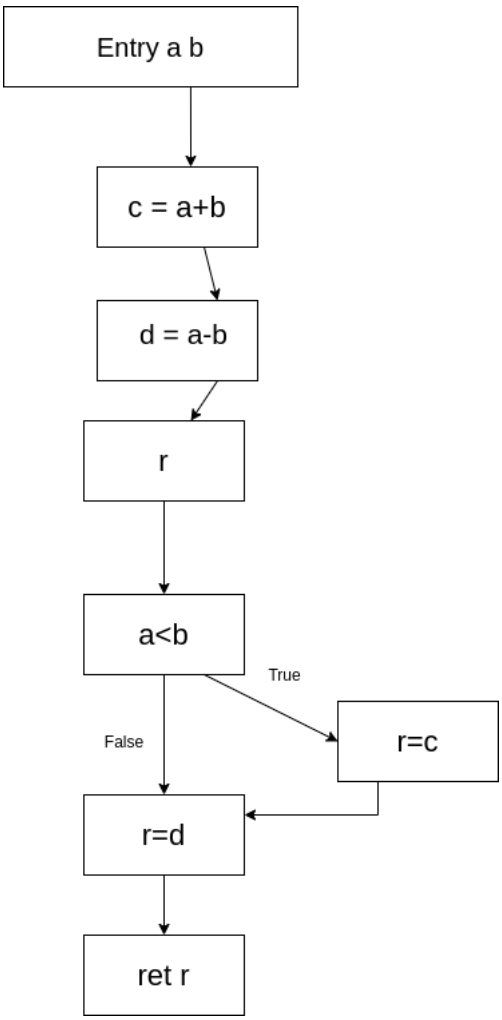

Figura 2: Control-Flow Graphs

2.2. ¿Cual es el conjunto de variables vivas en la lınea #5?

En base a la definición de este análisis el conjunto de variables vivas para IN y OUT es el siguiente:

IN	OUT
c	r

Para el análisis de Live Variables a diferencia de Reaching este tipo de análisis es BACKWARD.Por lo tanto tendremos dentro de nuestro conjunto de variables vivas como entrada en la linea 5 a la variable c y como salida la variable r, ya que en esta linea se utiliza la variable c y r donde r pasa a tener el valor de c.

Realizando un análisis mas exhaustivo para entender por que tenemos la variable c como entrada, debemos retraernos a la linea 1, en la cual se obtiene como out la variable c. Luego, las variables a y b no están vivas ya que las mismas no están siendo usadas.

2.3. ¿Cual es el conjunto de variables vivas en la lınea #7?

El conjunto de variables vivas IN OUT para este inciso es el siguiente:

IN	OUT
d	r

Análogamente al inciso anterior en lo que respecta al análisis, podemos notar que la variable dentro del conjunto IN es la variable d, ya que la misma es usada para realizar la operación **r** = **d**. Luego, al igual que en el inciso anterior la variable viva para el conjunto OUT es r.

2.4. ¿Cual es el conjunto de variables vivas en la lınea #9?

Por ultimo, para este inciso obtenemos el siguiente conjunto:

IN	OUT
r	$\emptyset$

Podemos observar que el conjunto OUT para este inciso es el $\emptyset$ dado que es el ultimo nodo que tiene alguna expresión en el cual se usan variables. Y para el conjunto IN, tomamos a la variable **r** ya que se esta usando en la expresión, y no se esta redefiniendo.

3. Ejercicio 3

A diferencia de los ítems anteriores en este punto se realiza un análisis **Null Pointer Checker** el cual encuentra las instrucciones que tienen el potencial de arrojar un **Null Pointer Exception**. Además, agrega anotaciones indicando, si el puntero siendo dereferenciado puede ser determinado estáticamente a un valor nulo.

3.1. ¿Que valores abstractos de las variables c1 y c2 pueden alcanzar la lınea 3?

En base al análisis, se confirma que no es posible que la expresión return c1.value arroje Null Pointer Exception. Si tomamos en cuenta los valores abstractos que podrían obtener las variables c1 y c2 notamos que los mismos serán *Not Null*.

Realizando un análisis detenidamente linea por linea del output de la herramienta con la que trabajamos , podemos ver que a las variables c1 y c2 se les asigna un valor *unknow* en lineas 1 y 2 respectivamente y en la linea 3 return c1.value nos devuelve un *Not Null*. Al estar c2 definida de forma idéntica a c1 puedo asumir que tendrá el mismo valor que c1 en caso de ser dereferenciada.

4. Readme

Para realizar la ejecución de cada análisis se realizo el siguiente procedimiento:

Trabajando con Linux Ubuntu:

- Se instalo Java versión 8
- Se descargo la herramienta SOOT provista por la cátedra
- Se seteo *jre* de la siguiente forma export JRE=/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/rt.jar (Para poder correrlo correctamente se tuvo que modificar /etc/-profile)
- Se compilo cada ejercicio por separado con el comando javac -g ex1.java
- Se ejecuto java -cp .:soot-3.3.0-jar-with-dependencies.jar:. soot.Main -cp .:JRE -f J ex1 -print-tags -p jap.rdtagger enabled:true -p jb use-original-names:true -p jb.cp off -keep-line-number para el analisis reaching definition.
- Analogamente para el ejercicio 2, se ejecuto luego de ser compilado el siguiente comando java -cp .:soot-3.3.0-jar-with-dependencies.jar:. soot.Main -cp . JRE -f J ex2 -print-tags -p jap.lvtagger enabled:true -p jb use-original-names:true -p jb.cp off -keep-line-number
- Por ultimo, para el ejercicio 3, se ejecuto luego de ser compilado el siguiente comando java -cp .:soot-3.3.0-jar-with-dependencies.jar:. soot.Main -cp . JRE -f J ex3 -print-tags -p jap.npc enabled:true -p jb use-original-names:true -p jb.cp off -keep-line-number