Ingeniería del Software II

Taller #7 – Implementando Análisis de Points-To

Deadline: Lunes 12 de Junio 23:59 hs
Fecha de Reentrega: Lunes 3 de Julio 23:59 hs

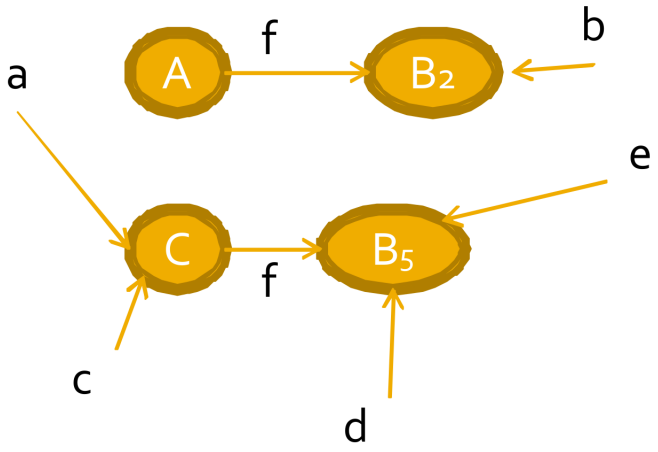
Introducción

Un análisis de *points-to* es un *may forward dataflow analysis* que permite determinar a qué objetos puede apuntar una variable durante la ejecución de un programa. En este taller vamos a desarrollar una versión *intra-procedural* de este algoritmo. Esto significa que vamos a asumir que el programa no tiene llamadas a funciones.

Para implementar el análisis utilizaremos un *points-to graph* (PTG) definido como $G = \langle N, E, L \rangle$ donde

- N es el conjunto de nodos del grafo, y representa los objetos creados por el programa. Cada nodo representa *todos* los objetos creados por una sentencia `new` del programa. Es decir, se creará un solo nodo incluso si el *new* se ejecuta dentro de un ciclo.
- E es el conjunto de aristas del grafo, y representa los objetos alcanzables desde cada objeto mediante campos. Una arista $e = (n_1, f, n_2)$ en el grafo indica que el nodo n_2 puede ser alcanzado desde el nodo n_1 mediante el campo f. Tener en cuenta que n_1 y n_2 representan ambos un conjunto de objetos.
- $L : Local \to P(N)$ es una función de etiquetado que asocia a cada variable local del programa un conjunto de nodos del grafo. Es decir, $L(x)$ es el conjunto de nodos a los que puede apuntar la variable x.

Gráficamente, un $PTG = \langle \{A, B_2, B_5, C\}, \{(A, f, B_2), (C, f, B_5)\}, [a \leftarrow \{C\}, b \leftarrow \{B_2\}, c \leftarrow \{C\}, d \leftarrow \{B_5\}, e \leftarrow \{B_5\}] \rangle$ luce de la siguiente forma:



El PTG debe ser utilizado como un reticulado por lo que hay que definir sus operaciones básicas:

- $\bot = \langle \{\}, \{\}, L \rangle$: con $L(x) = \{\}$ para toda variable local.
- $G_1 \subseteq G_2$ si $L_1 \subseteq L_2$ ó $(L_1 = L_2 \land N_1 \subseteq N_2)$ ó $(L_1 = L_2 \land N_1 = N_2 \land E_1 \subseteq E_2)$.
- $G_1 \cup G_2 = \langle N_1 \cup N_2, E_1 \cup E_2, L_1 \cup L_2 \rangle$.

Las reglas Dataflow para el análisis son las siguientes:

Expresión	**Efecto**
$p : x = newA()$	$G' = G$ con $L'(x) = \{p\}$, donde p es el número de línea en el programa.
$x = y$	$G' = G$ con $L'(x) = L(y)$
$x = y.f$	$G' = G$ con $L'(x) = \{n_2 \mid (n_1, f, n_2) \in E \wedge n_1 \in L(y)\}$
$x.f = y$	$G' = G$ con $E' = E \cup \{(n_1, f, n_2) \mid n_1 \in L(x) \wedge n_2 \in L(y)\}$

Pueden asumir que no hay variables globales.

Taller

Para la implementación del algoritmo vamos a utilizar la librería *SOOT* que provee una representación intermedia de los programas en Java. Para este objetivo tendrán que completar una implementación ya existente que deberán bajar del campus de la materia. Al importar el proyecto en *IntelliJ IDEA*, puede hacer falta configurar el JDK de la siguiente manera: Settings > Build, Execution, Deployment > Build Tools > Gradle → Gradle JVM 1.8

Deberán completar todos los métodos marcados con `TODO`.

Para correr los tests del proyecto puede hacerlo desde una IDE como *IntelliJ IDEA* o desde la terminal con el comando `./gradlew test`. Si se desea correr el Points-to Analysis desde la terminal para una clase, se puede hacer con el comando `./gradlew pointsToAnalysis <targetClass>`.

Recordar que para que este taller funcione se debe tener instalado Java SE Runtime Environment 8 (ver Taller anterior ante dudas).

Formato de Entrega

El taller debe ser subido al campus. Debe ser un archivo zip con el siguiente contenido.

1. Una carpeta con la implementación completa y funcionando del Points-to Analysis. Esta implementación debe pasar los tests del archivo `TallerTest.java`. Para los métodos modificados, agregue comentarios explicando la solución desarrolada.