Trabajo Práctico 1

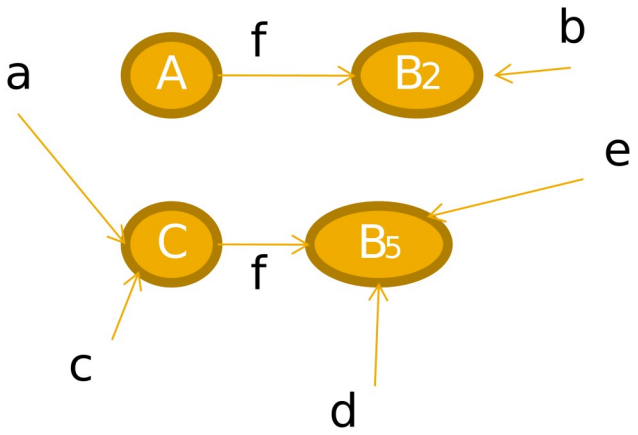
Análisis Dataflow – Points-to analysis

El objetivo del trabajo práctico es desarrollar un análisis de poinst-to para Java utilizando el framework de Dataflow de soot. Para ello deberán primero desarrollar una versión intra-procedural, para luego extender el análisis a varios procedimientos.

Para implementar en análisis utilizaremos un points-to graph (PTG) definido de la siguiente manera: **G = <L , N, E, W>** donde

- **N** representa los objetos creados por el programa. Cada nodo representa todos los objetos creados en una sentencia **new**. Es decir, se creará un solo nodo incluso si el new se produce dentro de un ciclo.
- **L**: Local → P(N) es un map de variables locales a nodos. Representa los objetos a los que puede apuntar cada variable local.
- **E** es un conjunto de ejes. Un eje *(n1, f, n2)* indica que un nodo *n1* (representando un conjunto de objetos) apunta a otro nodo n2, mediante el campo *f*.
- **W** es un conjunto de sentencias: representa los puntos donde el análisis tuvo problemas (ver punto 1)

Graficamente, un PTG = <[a←{C},b←B2,c←C,d←B5,e←B5], { (A,f,B2), (C,f,B5)}, {A,B2,B5,C}> luce de la siguiente forma:



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

- $\perp$ = <L, {}, {}> con L(x)={} para toda variable local
- G1 $\subseteq$ G2 si L1$\subseteq$L2 ó (L1=L2 y N1$\subseteq$N2) ó (L1=L2 yN1=N2 y E1$\subseteq$E2)
- G1$\cup$G2 = < L1$\cup$L2, N1$\cup$N2, E1$\cup$E2>

Utilizaremos un lenguaje Java hiper simplificado. Las reglas Dataflow para el análisis son las siguientes:

Sentencia	Efecto
p: x = new A()	G' = G con L'(x) = {A_p}
x = y	G' = G con L'(x) = L(y)
x = y.f	G' = G con L'(x) = { n | (a,f,n) in E forall *a* in L(y)}
x.f = y	G' = G con E' = E U { (a,f,n) | n in L(y) && *a* in L(x) }
ret = a0.m(a1,..,an)	Ver parte interprocedural

Pueden asumir que no hay variables globales.

Parte 1: Análisis intraprocedural

1) Desarrollar un análisis intraprocedural siguiendo las reglas mencionadas en la tabla. En el caso de llamadas a métodos, no analizarlas pero incluir la sentencia en el conjunto W (no analizables).
2) Extender el análisis intraprocedural de forma tal que pueda distinguir "lecturas" de "escrituras" y soportar parámetros. Para ello se debe incluir un nuevo tipo de nodo que represente a cada unos de los parámetros del método (parameter nodes). Estos deben ser ligados a los parámetros del método al comienzo de la ejecución del mismo (L(p)={PN} para todo p). Además agregaremos un nuevo tipo de nodo llamado load node (ln) y un nuevo tipo de ejes de la forma (n,f,ln) (conjunto R de ejes). La nuevas reglas son las siguientes:

Sentencia	Efecto
p: x = new A()	G' = G con L'(x) = {A_p}
x = y	G' = G con L'(x) = L(y)
x = y.f	G' = G con R' = R U { (n,f,ln) | n in L(y) } y L'(x) = { ln} con ln fresco
x.f = y	G' = G con E' = E U { (a,f,n) | n in L(y) && *a* in L(x) }
ret = a0.m(a1,..,an)	Ver parte interprocedural

Parte 2: Análisis interprocedural

3) Realizar un análisis interprocedural del ejercicio 1 utilizando la técnica del **CFG interprocedural**. Es decir conectando de forma adecuada los diferentes CFG de los métodos del programa bajo análisis
4) Realizar un análisis interprocedural simulando la técnica de **inlining**. Es decir, utiliizando un solo PTG global, pero cada vez que se realiza una llamada a un método, simular que el método llamado es parte del método original. Para ello deben realizar el correcto *binding* de parámetros de entrada y salida. En el caso de recursión pueden incluir el método en el conjunto de no analizables.
5) Utilizar el resultado del análisis para mostrar el grafo de llamadas del programa analizado.

6) (opcional) Extender el análisis utilizando la técnica de **call strings** o **K-limiting**, limitando el contexto de llamada a **K** métodos (un valor que pueda ser configurable). Para implementar está técnica se puede tanto la idea de limitar los caminos disponibles en el CFG interprocedural o también se puede simular este efecto.
   Por ejemplo, se podría guardar un diccionario de call string a la información de Dataflow de cada método. De esta forma se podría recuperar bajo demanda la información de Dataflow de un método para un determinado contexto (dado por el call string) y calcular a partir de esa información utilizando el algoritmo intraprocedural.
7) (super-opcional) Realizar una versión **interprocedural** utilizando resúmenes. Los resúmenes tendrán la información de points-to al final del método (borrando la información de variables locales). Para ello se debe incluir un nuevo tipo de nodo que represente a cada unos de los parámetros del método. Luego a la hora del realizar el call se deben "pegar" el PTG del llamador con el del llamado. Esto se logra tomando el resumen del llamado y fusionado los nodos que representan los parámetros con los nodos que representan a los argumentos del llamador. De esta manera, el PTG del llamador al final del call tendrá la información de points-to del m{etodo llamado.

Observación: Para el análisis interprocedural van a tener que decidir qué método (de que clase) es llamado en una invocación. Para eso tienen que usar el mismo análisis de points-to que estan calculando.

Observación 2: En el análisis interprocedural **no deben analizar las librerías de Java**. Para eso pueden incluir las sentencias con llamadas a funciones de la librería Java en el conjunto **W**.