# Ingeniería del Software II

Práctica #1 – Análisis Estático

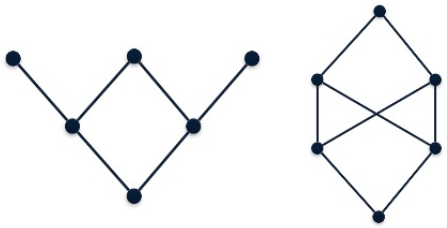
**Parte 1 – Reticulados, Punto fijo**

**Ejercicio 1**

1. Definir un reticulado usando el orden parcial $\langle \mathbb{Z}, \leq \rangle$. Existen elementos $\top$ y $\bot$?
2. Definir un reticulado completo sobre el conjunto $\mathbb{Z} \bigcup \{-\infty, \infty\}$.

**Ejercicio 2**

Sean los siguientes órdenes parciales:



1. Indicar si son reticulados.
2. En caso que no lo sean, indicar por qué no lo son y cómo los repararían.

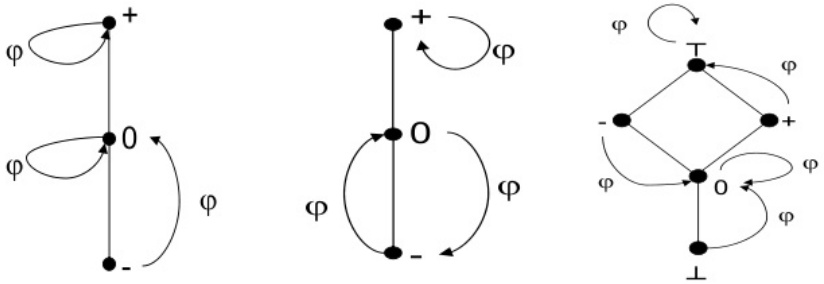
**Ejercicio 3**

Sea $S = \{'A', 'B', 'C'\}$

1. Armar un reticulado de altura 2 donde los elementos de $S$ no sean comparables
2. Armar el retuculado de partes de $S$. Que altura tiene?
3. Sea $N = \{1, 2\}$. Armar el reticulado de pares de $S$ y $N$.

**Ejercicio 4**

Sean las siguientes funciones:




1. Calcular $\varphi(\sqcup X)$ para todo $X \subseteq S$

2. Enumerar todas las cadenas incrementales (*increasing chains*) en $S$
3. Es $\varphi$ monótona?
4. Computar iterativamente el lfp (mínimo punto fijo) de $\varphi$.

**Ejercicio 5**

El teorema del punto fijo también se cumple sin la suposición de que f es monótona? Si es así, da una prueba; si no, da un contraejemplo.

**Ejercicio 6**

Sea el reticulado de signo (i.e.,$Lift(\{-, 0, +\})$):

1. Extenderlo para incluir 2 símbolos que representen los valores mayores iguales a cero y los menores que cero.
2. Definir la operación suma en el nuevo reticulado.

**Ejercicio 7**

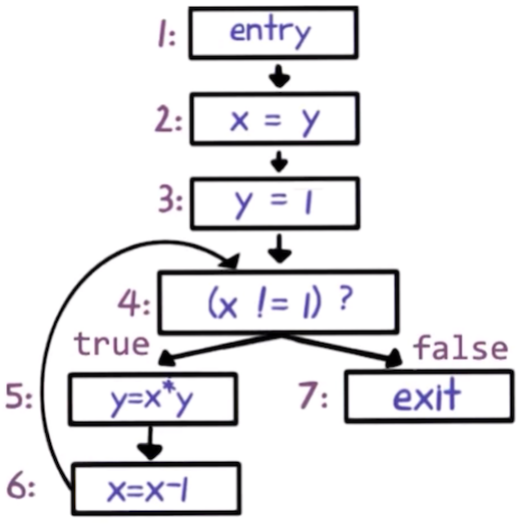
Sea el conjunto que representa funciones de variables a signos. $D : Var \mapsto Sign$.

1. Definir la operación $\sqsubseteq$ que determine que dados $d1, d2 \in D$ $d1 \sqsubseteq d2$ si $d1$ es igual o más preciso que $d2$.
2. Definir la operacion $\sqcup$ que represente el supremo en $D$.
3. Dada la function $x+ = E$ que dado un elemento $d \in D$ , $x+ = E(d) = [x \mapsto x + sign(E)]$, donde $sign(E)$ es el valor de $E$ en el reticulado $Sign$. Es $x+ = E$ monótona? Justificar.

**Parte 2 – Dataflow análisis**

**Ejercicio 8**

Sea el siguiente control-flow graph para una función:



Ejecutar el algoritmo caótico iterativo para el análisis de Reaching Definitions hasta alcanzar la estabilidad de los conjuntos IN y OUT. Completar la siguiente cuadro con el valor final de los conjuntos IN y OUT:

| Nodo n | IN[n] | OUT[n] |
|---|---|---|
| 1 | - | $\emptyset$ |
| 2 | $\emptyset$ | $\{\langle x, 2\rangle\}$ |
| 3 | $\{\langle x, 2\rangle\}$ | $\{\langle x, 2\rangle, \langle y, 3\rangle\}$ |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

**Ejercicio 9**

Sean los conjuntos KILL[n] y GEN[n] para el análisis de Reaching Definitions (*) que matan y generan respectivamente la información de dataflow, completar las ecuaciones de dataflow que caracterizan los conjuntos IN[n] y OUT[n] para dicho análisis.

(*) Recordar que un análisis Dataflow, se utilizan las funciones de tranferencia para indicar la semantica de cada instrucción. Estas se suelen definir diciendo lo que se "mata" y lo que se "genera" en la instrucción que se analiza. La función de transferencia se define en el IN o en el OUT dependiendo del tipo de analisis (ej: Forward, Backward, May, MUST). En particular, dependiendo del tipo de análisis el IN[n] o el OUT[n] para un nodo n, se puede definir usando KILL y GEN.

**Ejercicio 10**

Sea el siguiente programa, donde `MASK`, `IA`, `IQ`, `IR`, `IM` y `AM` son constantes.

```
float foo(int pid) {
1:  int i, j, h;
2:  i = pid ^ MASK;
3:  int k = i / IQ;
4:  h = IA * (i - k * IQ) - IR * k;
5:  h = j ^ MASK;
6:  if (h < 0)
7:    h = h + IM;
8:  float answer = AM * h;
9:  return answer * pid / k;
}
```

a) Construir su control-flow graph.
b) Computar el análisis Live Variables.

**Ejercicio 11**

Computar los conjuntos IN y OUT para el análisis de Available Expressions.

```
void foo(int[] m) {
1: int a = 3;
2: int i = 0;
3: while (i <= a) {
4:   int t = m[i];
```

```
5:     m[i] = t;
6:     i = i + 1;
    }
8:  bar(M, a);
}
```

**Ejercicio 12**

Categorizar los análisis dataflow Live Variables, Reaching Definitions, Very Busy Expressions y Available Expressions.

| | ***Forward*** | ***Backward*** |
|---|---|---|
| ***May*** | | |
| ***Must*** | | |

**Ejercicio 13**

Sea el siguiente programa:

```
1  void bar(int p1, int p2) {
2    int a = p1;
3    int b = p2;
4    int x = a+b;
5    int y = a*b;
6    while (y>a) {
7      a = a +1;
8      x = a+b;
9    }
10   return;
11 }
```

a. Escribir el control-flow graph del programa

b. Ejecutar el algoritmo caótico iterativo para el análisis de Live Variables hasta alcanzar la estabilidad de los conjuntos IN y OUT y completar la siguiente tabla:

| Nodo n | IN[n] | OUT[n] |
|---|---|---|
| ... | ... | ... |

**Ejercicio 14**

Supongamos que tenemos programas que manipulan datos sensibles. Estos datos se encuentran en variables.

Usando la función `sensible(x)` decimos que el dato en la variable `x` es sensible y con la función `insensible(x)` decimos que `x`no es sensible. Un dato sensible puede pasare a otra variable si es leido. Por ejemplo `y = x + 1` hace que `y` sea sensible porque leyo un dato sensible que venia de `x`.

1. Definir un reticulado que modele el uso de variables sensibles e insensibles.

2. Definir un análisis Dataflow para calcular sensibilidad de variables en cada punto del programa. Es decir para cada nodo $n$ del CFG, OUT$[n](x)$ debería indicar si $x$ es sensible o no.

3. Calcular en análisis para el siguiente programa.

```
1 x = 1;
2 sensible(x);
3 y = input;
4 if(y>0)
5   z = x + 1;
6 else
7   z = 0
8 insensible(x);
```

**Ejercicio 15**

Sea el siguiente programa:

```
1 y = 0;
2 x = 0;
3 while (true) {
4   x = y +1;
5   if (y==0)
6     x = 0;
7 }
```

1. Constuir el CFG del programa.
2. Calcular las igualdades de la forma $x = y$.
   - Definir un reticulado para trackear igualdad entre variables (hint: usar conjunto de pares de variables)
   - Definir la funciones de transferencia para las instrucciones que aparecen en el programa.
   - Definir las ecuaciones de Dataflow para el programa.
   - Calcular la soluciıon a las ecuaciones (usando cualquier algorirmo de punto fijo)

**Ejercicio 16**

Sobre el mismo programa del ejercicio anterior. Computar el signo de todas las variables (definiendo el análisis de dataflow correspondiente)