Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 4
Cobertura basada en grafos

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

24 de agosto de 2020

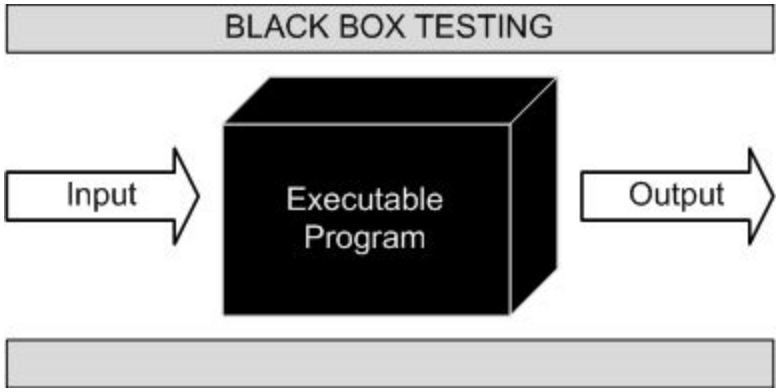
Encuestas

- Conocimientos generales
- Grupos proyectos semestral

1. Clase pasada
    - Motivación
    - Conceptos
    - Tipos de *tests*
    - Cobertura
2. Criterios de cobertura
    - Cobertura basada en grafos

Black-box testing

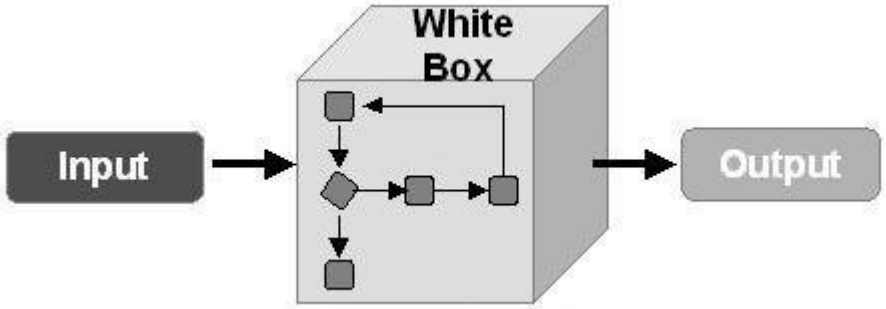
- No se conocen los detalles del *software*, solamente se considera *input* y *output*.



- Ejemplos
  - Pruebas de usuarios
  - Pruebas de seguridad

White-box testing

- Se conocen los detalles del software y se puede utilizar al diseñar los *tests*



Black-box vs White-box

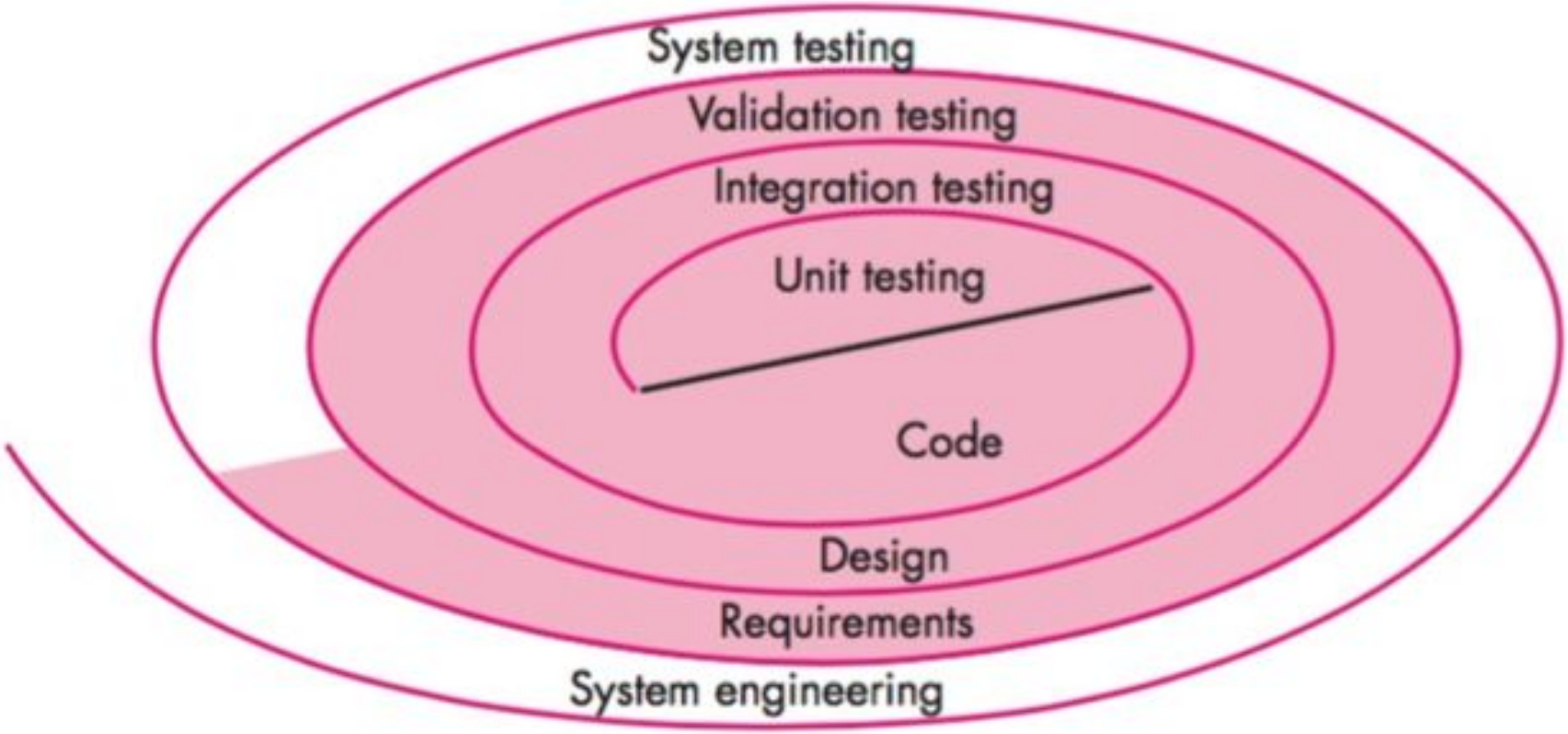
- *Black-box*

```
// Retorna la cantidad de ceros contenidos en un arreglo
function countZeros(array) {
```

- *White-box*

```
// Retorna la cantidad de ceros contenidos en un arreglo
function countZeros(array) {
  var count = 0;
  for (var i = 1; i < array.length; i++) {
    if (array[i] === 0) {
      count++;
    }
  }
  return count;
}
```

Niveles tradicionales de *Testing*



Tests de humo

- Subconjunto de pruebas enfocadas en garantizar las funcionalidades más importantes

- Se ejecutan de manera rápida y barata antes de cada implementación

Tests de mutación

- Se introducen pequeñas variaciones en el código con el objetivo de cuantificar cuántos *tests* fallan
- Evalúan la calidad de los *tests* existentes

```
# Original
if x > y:
    z = x - y
else:
    z = 2 * x
```

```
# Mutación 1
if x >= y:
    z = x - y
else:
    z = 2 * x
```

```
# Mutación 2
if x > y:
    z = x + y
else:
    z = 2 * x
```

```
# Mutación 3
if x > y:
    z = x - y
else:
    z = 2 * y
```

Tests de regresión

- *Tests* para garantizar que luego de modificar un software las funcionalidades originales siguen respetando las especificaciones
- Pueden ser un subconjunto de las pruebas o la batería completa
- Pueden ser útiles al momento de versionar código según el criterio SemVer

Cobertura (*coverage*)

- Una métrica que mide la proporción de código fuente que son probadas por una batería de *tests*.

- Existen distintos criterios de cobertura de pruebas.

Criterios de cobertura de pruebas

- Las pruebas son caras y consumen esfuerzo
- Los criterios de cobertura sirven para decidir qué entradas de prueba usar
- Los criterios hacen que las pruebas sean más eficientes y efectivas:
  - Es más probable encontrar problemas
  - Generan mayor confianza en la calidad del código
  - Se responde al por qué de cada prueba

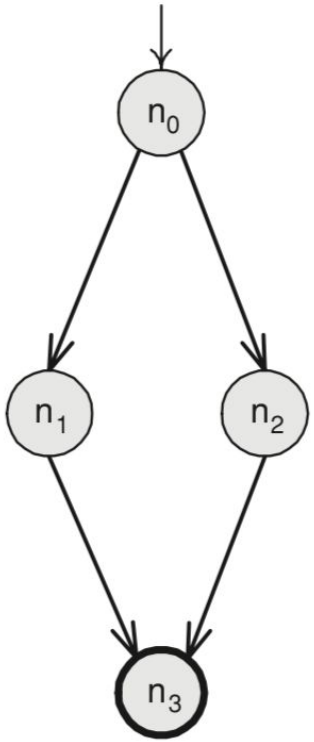
Criterios de cobertura de pruebas

- Basados en grafos
- Expresiones lógicas
- Dominio de parámetros de entrada
- Estructuras sintácticas

Grafo

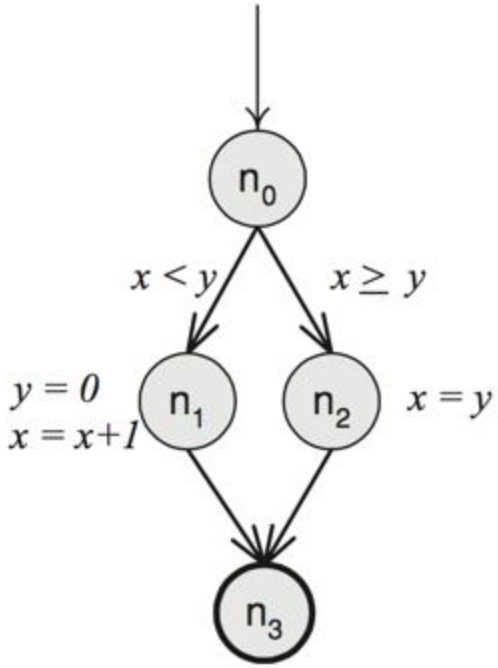
- **N**: Set de nodos
- $\mathbf{N_0}$: Set de nodos iniciales, subconjunto de **N**
- $\mathbf{N_f}$: Set de nodos finales, subconjunto de **N**
- **E**: Set de aristas, subconjunto de **N x N**

Grafo: Ejemplos

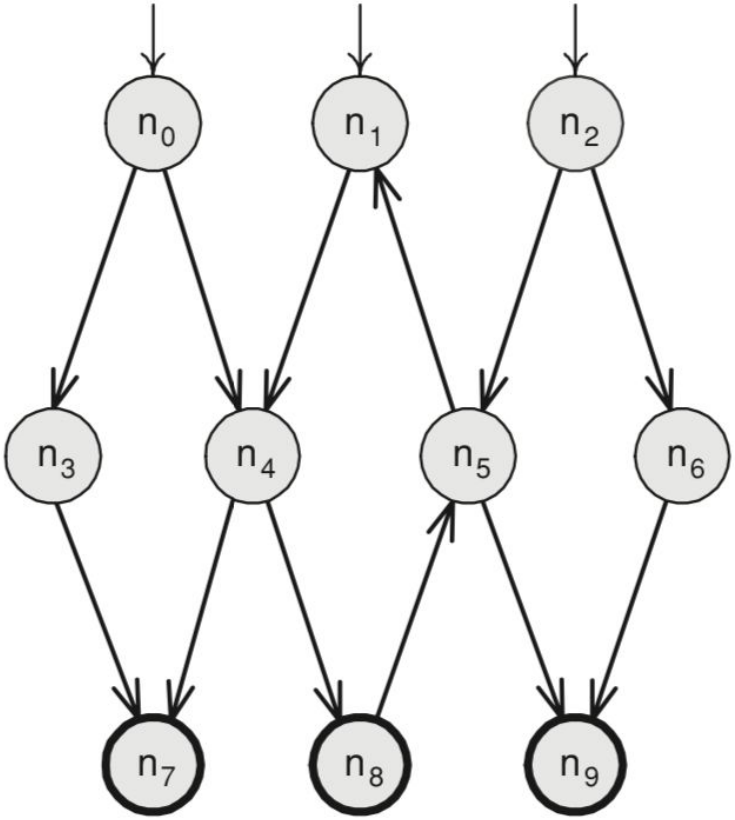


$N = \{ n_0, n_1, n_2, n_3 \}$
$N_0 = \{ n_0 \}$
$E = \{ (n_0, n_1), (n_0, n_2), (n_1, n_3), (n_2, n_3) \}$

```
if (x < y)
{
   y = 0;
   x = x + 1;
}
else
{
   x = y;
}
```



Grafo: Ejemplos



$N = \{ n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9 \}$

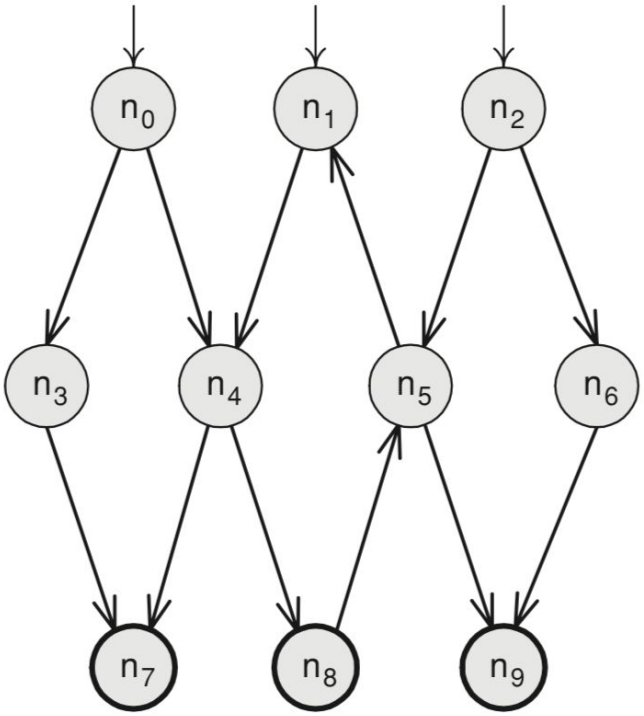
$N_0 = \{ n_0, n_1, n_2 \}$

$|E| = 12$

Grafo: Definiciones

Camino:

- secuencia de nodos donde cada par de nodos adyacentes ($\mathbf{n_i}$, $\mathbf{n_{i+1}}$) está contenido en el set **E** de aristas.



Path Examples	
1	n_0, n_3, n_7
2	n_1, n_4, n_8, n_5, n_1
3	n_2, n_6, n_9

Invalid Path Examples	
1	n_0, n_7
2	n_3, n_4
3	n_2, n_6, n_8

Grafo: Definiciones

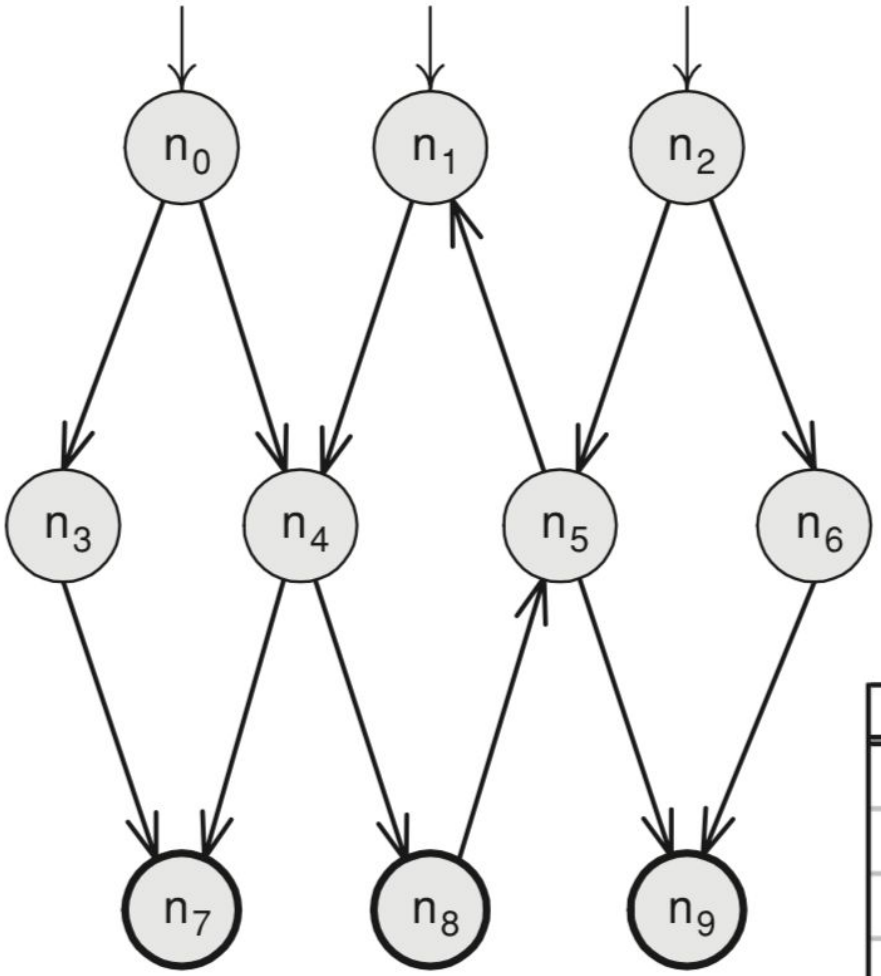
- *Camino*: secuencia de nodos donde cada par de nodos adyacentes ($\mathbf{n_i}$, $\mathbf{n_{i+1}}$) está contenido en el set **E** de aristas.

- *Sub-camino*: subsecuencia de nodos de un camino **p**.

  $[\mathbf{n_0}, \mathbf{n_3}]$ es subcamino de $[\mathbf{n_0}, \mathbf{n_3}, \mathbf{n_7}]$

- *Largo de camino*: cantidad de aristas contenidas en el camino.

Alcance Sintáctico y Semántico



¿Es $\mathbf{n_2}$ alcanzable desde $\mathbf{n_0}$?

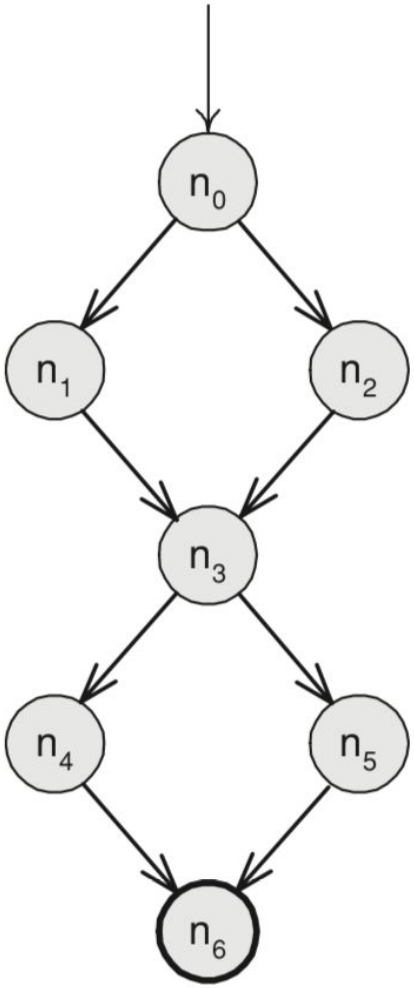
¿Es $\mathbf{n_7}$ alcanzable desde $\mathbf{n_1}$?

Sintáctico: Depende de la estructura del grafo.

Semántico: Depende la semántica del software.

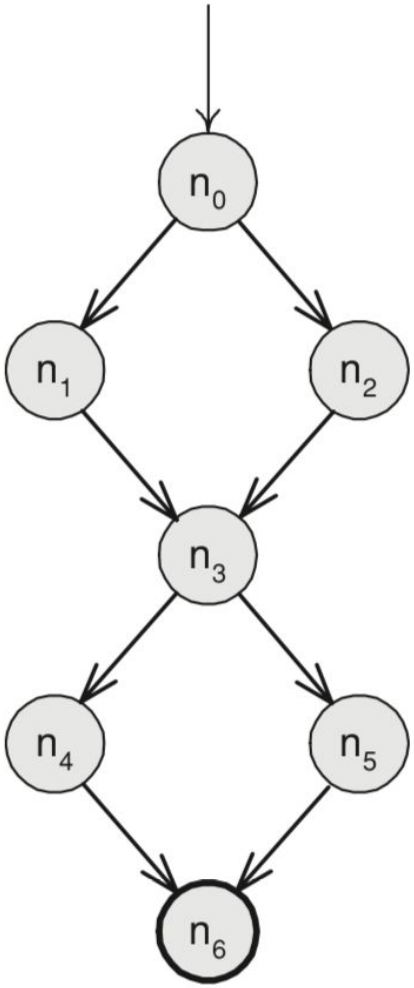
Reachability Examples	
1	*reach* $(n_0) = N - \{n_2, n_6\}$
2	*reach* $(n_0, n_1, n_2) = N$
3	*reach* $(n_4) = \{n_1, n_4, n_5, n_7, n_8, n_9\}$
4	*reach* $([n_6, n_9]) = \{n_9\}$

Grafo **S**ingle **E**ntry **S**ingle **E**xit (**SESE**)



- $|\mathbf{N_0}| = 1$
- $|\mathbf{N_f}| = 1$
- reach($\mathbf{n_0}$) = **G**
- reach($\mathbf{n_f}$) = [$\mathbf{n_f}$]

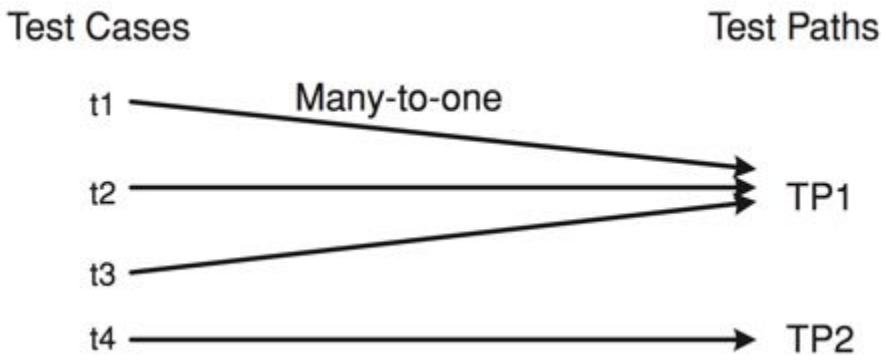
Camino de prueba (test path)



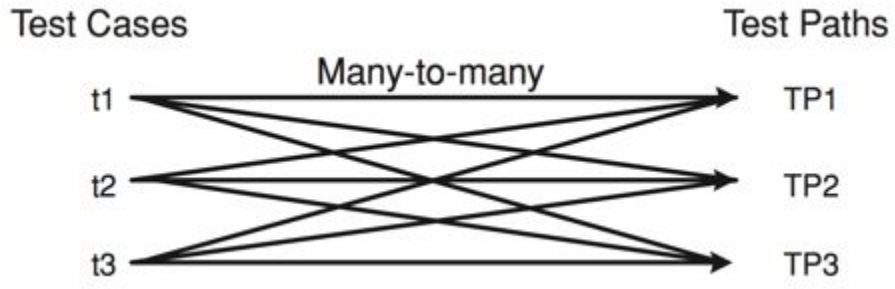
- Es un camino **p** que comienza en algún nodo de $\mathbf{N_0}$ y termina en algún nodo de $\mathbf{N_f}$.
- Corresponde a la ejecución de uno o varios casos de pruebas.

$$\mathbf{t_1} = [\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_3}, \mathbf{n_4}, \mathbf{n_6}]$$
$$\mathbf{t_2} = [\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_3}, \mathbf{n_5}, \mathbf{n_6}]$$
$$\mathbf{t_3} = [\mathbf{n_0}, \mathbf{n_2}, \mathbf{n_3}, \mathbf{n_4}, \mathbf{n_6}]$$
$$\mathbf{t_4} = [\mathbf{n_0}, \mathbf{n_2}, \mathbf{n_3}, \mathbf{n_5}, \mathbf{n_6}]$$

Casos y Caminos de prueba

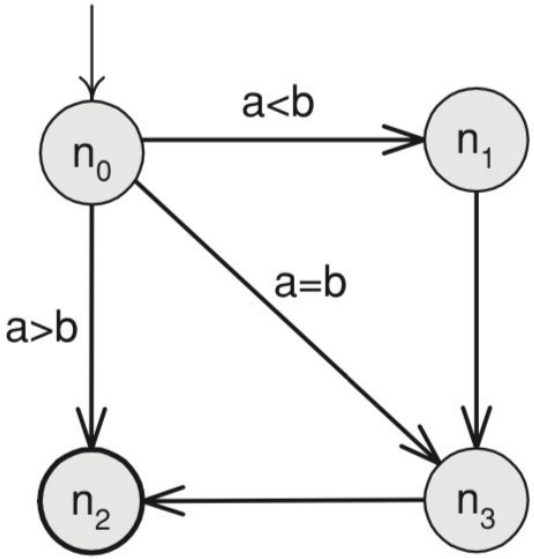


Software Determinístico



Software No Determinístico

Casos y Caminos de prueba



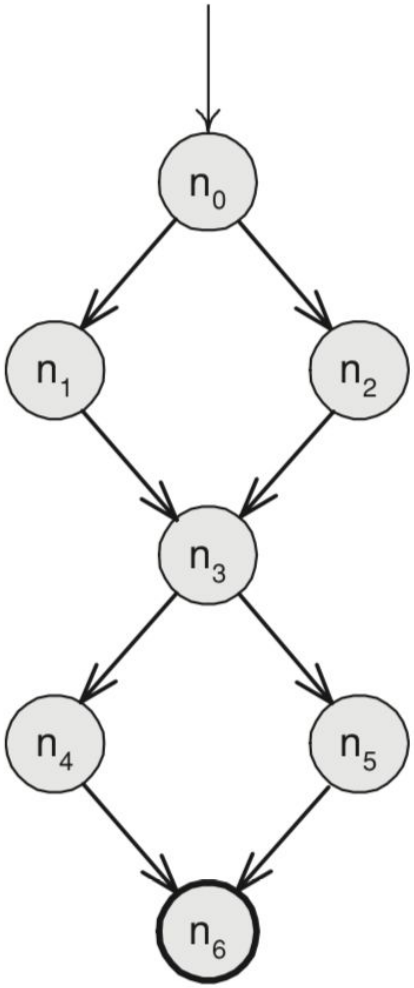
Map to

Test case t_1 : (a=0, b=1) → [Test path p_1 : n_0, n_1, n_3, n_2]

Test case t_2 : (a=1, b=1) → [Test path p_2 : n_0, n_3, n_2]

Test case t_3 : (a=2, b=1) → [Test path p_3 : n_0, n_2]

Visitar y Recorrer



- **p** = [$\mathbf{n_0}$, $\mathbf{n_1}$, $\mathbf{n_3}$, $\mathbf{n_4}$, $\mathbf{n_6}$]
- **p** "visita" el nodo **n** si $\mathbf{n} \in \mathbf{p}$
- **p** "visita" la arista **e** si $\mathbf{e} \in \mathbf{p}$
- **p** "recorre" el sub-camino **q** si $\mathbf{q} \in \mathbf{p}$

Criterios y Requerimientos de prueba

Criterio de prueba (**C**):

- reglas que definen requerimientos de prueba.

Ej: “Visitar todos los nodos”

Requerimiento de prueba (**TR**):

- describe propiedades de un camino de prueba.

Ej: “Visitar $\mathbf{n_0}$”

Criterios y Requerimientos de prueba

"Dado un conjunto de requerimientos de prueba ***TR*** para un criterio de cobertura ***C***, un conjunto de pruebas ***T*** satisface ***C*** si y sólo si para cada requerimiento ***tr*** en ***TR*** hay al menos un camino de prueba ***p*** que satisface ***tr***."

Criterios de cobertura en grafos

Los criterios de cobertura en grafos se dividen en 2 tipos:

- Estructural
- Flujo de información

Estructural

- En general hacen referencia a visitar nodos o aristas según un criterio específico.

- Se utilizará la siguiente notación:

$$\mathbf{TR} = \{\text{visita } \mathbf{n_0}, \text{visita } \mathbf{n_1}\}$$

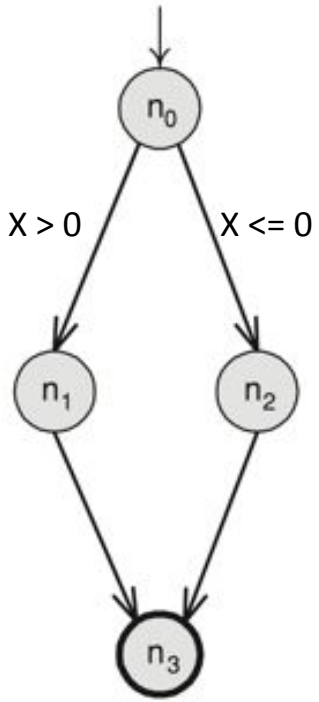
$$\mathbf{TR} = \{\mathbf{n_0}, \mathbf{n_1}\}$$

NC: Cobertura de nodos

(Node Coverage o Statement Coverage)

- Criterio:

TR contiene todos los nodos alcanzables de **G**.



$\mathbf{TR} = \{\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}\}$

$\mathbf{p_1} = [\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_3}]$
$\mathbf{p_2} = [\mathbf{n_0}, \mathbf{n_2}, \mathbf{n_3}]$

T = $\{\mathbf{t_1}, \mathbf{t_2}\}$ tal que:

- path($\mathbf{t_1}$) = $\mathbf{p_1}$
- path($\mathbf{t_2}$) = $\mathbf{p_2}$

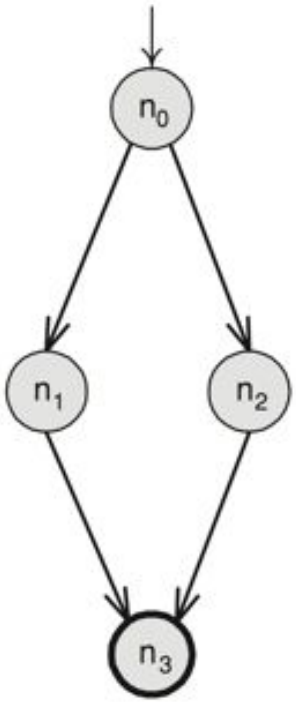
$\mathbf{t_1}$ = {x=1}
$\mathbf{t_2}$ = {x=0}

EC: Cobertura de aristas

(Edge Coverage o Branch Coverage)

- Criterio:

TR contiene todos los caminos de largo 1 alcanzables de **G**



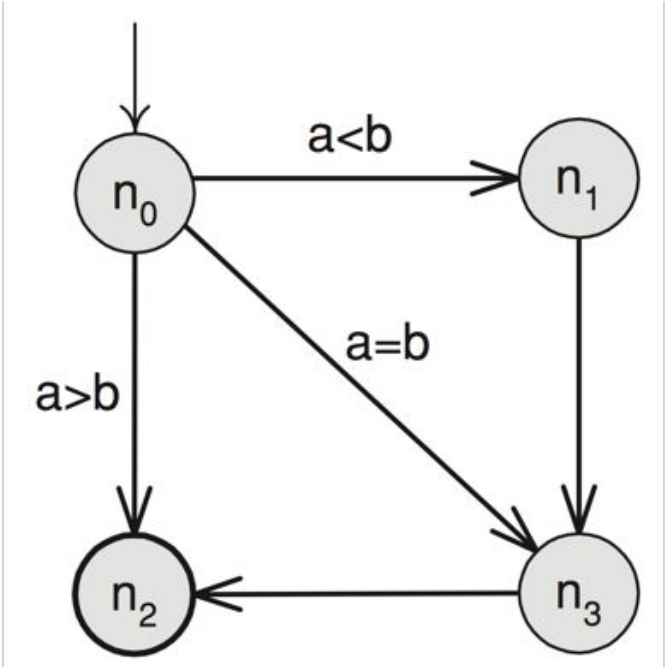
$$\mathbf{TR} = \{(\mathbf{n_0},\mathbf{n_1}) ; (\mathbf{n_1},\mathbf{n_3}) ; (\mathbf{n_0},\mathbf{n_2}) ; (\mathbf{n_2},\mathbf{n_3})\}$$

$$\mathbf{p_1} = [\mathbf{n_0}, \mathbf{n_1}, \mathbf{n_3}]$$
$$\mathbf{p_2} = [\mathbf{n_0}, \mathbf{n_2}, \mathbf{n_3}]$$

T = {$\mathbf{t_1}$, $\mathbf{t_2}$} tal que:

- path($\mathbf{t_1}$) = $\mathbf{p_1}$
- path($\mathbf{t_2}$) = $\mathbf{p_2}$

NC vs EC



NC:

$p_1 = [n_0, n_1, n_3, n_2]$

EC:

$p_1 = [n_0, n_2]$

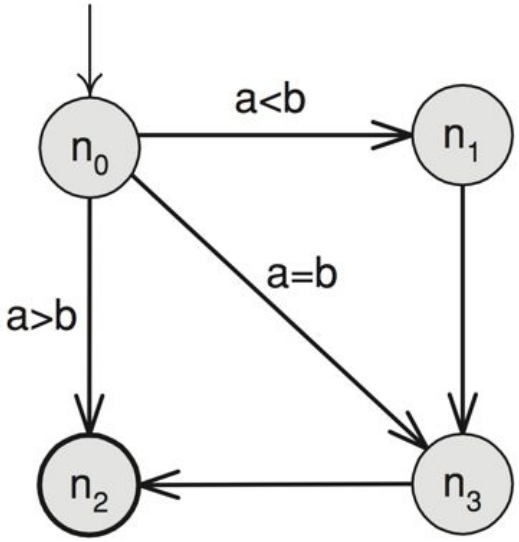
$p_2 = [n_0, n_3, n_2]$

$p_3 = [n_0, n_1, n_3, n_2]$

EPC: Cobertura de pares de aristas
(Edge Pair Coverage)

- Criterio:

TR contiene todos los caminos de largo hasta 2 alcanzables de **G**



$$\mathbf{TR} = \{(n_0,n_1,n_3) ; (n_0,n_3,n_2) ; (n_1,n_3,n_2) ; (n_0,n_2) ; (n_0,n_1) ; (n_0,n_3) ; (n_1,n_3) ; (n_3,n_2)\}$$

$$p_1 = [n_0, n_1, n_3, n_2]$$
$$p_2 = [n_0, n_3, n_2]$$
$$p_2 = [n_0, n_2]$$

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 4
Cobertura basada en grafos

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

24 de agosto de 2020