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

Clase 14
Cobertura en base a dominio

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

19 de octubre de 2020

Criterios de cobertura de pruebas

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

Dominio de parámetros de entrada

- Los casos de prueba son básicamente entradas para cubrir ciertos escenarios de ejecución del *software*
  - Existen distintas entradas con resultados similares
- Al analizar los parámetros:
  - No se necesita saber nada de la implementación
  - Se puede ajustar para generar más/menos pruebas
  - Puede ser aplicado en distintos niveles de pruebas

Dominio de parámetros de entrada

- Problemas
  - ¿Cómo sabemos qué entradas considerar?
  - ¿Cómo sabemos qué casos son similares a otros?

Definiciones

- **Dominio de entradas (*Input domain*)*:*** contiene todos las posibles entradas para un programa
  - Incluso en programas pequeños, el dominio de entradas es conceptualmente infinito
- **Parámetro de entrada (*Input parameter*)*:*** cualquier tipo de entrada del programa
  - Archivos, variables globales, lecturas de periféricos, parámetros para un método, etc..

Definiciones

- El dominio de entrada se forma con las combinaciones de los dominios particulares de parámetros de entrada

- Formalmente, el dominio de entrada es el producto cruz de los dominios de cada parámetro de entrada

Partición de dominio de entradas

Input Space Partition

- Debemos elegir un conjunto finito del dominio de entrada
  - Se particiona el dominio de entrada en regiones "equivalentes"
  - Se selecciona a lo menos un valor de cada región

Definición: Partición

- Dado un dominio ***D***, una partición ***q*** define un conjunto de bloques B_q tal que:
  - Los bloques son disjuntos
    - ningún par de bloques se traslapa

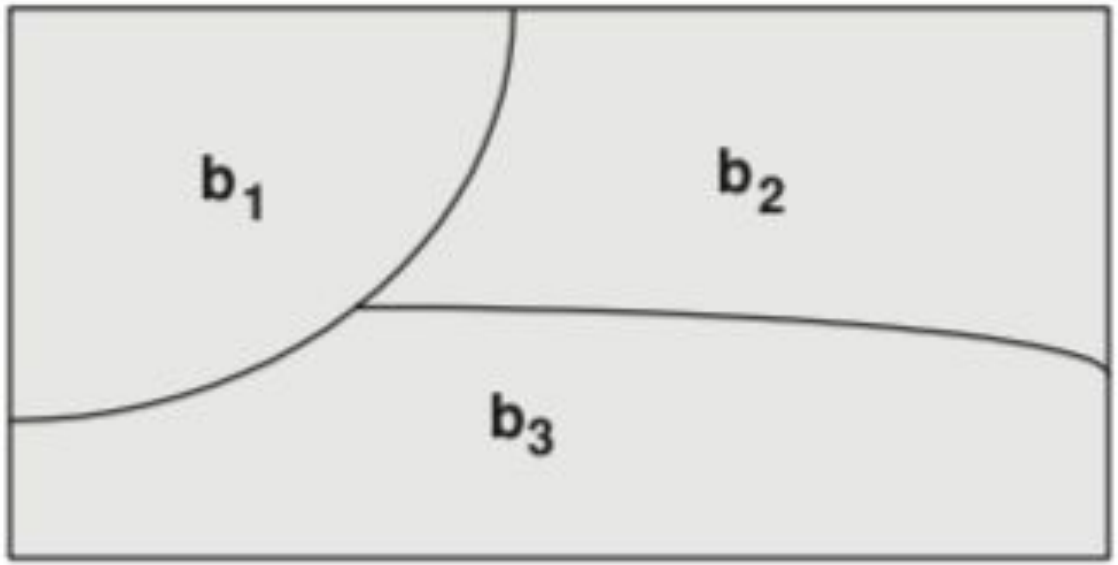
$$b_i \cap b_j = \emptyset, i \neq j; b_i, b_j \in B_q$$

  - Los bloques con completos
    - el conjunto de bloques debe cubrir el dominio D

$$\bigcup_{b \in B_q} b = D$$

- Las particiones se basan en características ***C*** del programa.

Definición: Partición



Partición: Ejemplo

- Característica:
  - c_1: orden de una lista de números
- Particiones:
  - b_1: ordenado ascendente
  - b_2: ordenado descendente
  - b_3: desordenado
- ¿Es una partición válida?
  - Las listas vacías o de largo 1 están contenidas en los 3 bloques.
  - No es una partición disjunta.

Partición: Ejemplo

- Características:
  - c_1: ordenado ascendente
  - c_2: ordenado descendente

- Particiones:
  - c_1
    - b_1: Sí
    - b_2: No
  - c_2
    - b_1: Sí
    - b_2: No

Definición de características

Existen 2 enfoques:

- Basado en la interfaz
- Basado en la funcionalidad

Enfoque basado en interfaz

- Se obtienen las características desde los parámetros individuales
- Es simple y automatizable
- Puede ser incompleto al no considerar parte de la información del dominio
  - Semántica
  - Relaciones entre los parámetros

Enfoque basado en funcionalidad

- Se obtienen características según el comportamiento esperado del *software*
- Es más costoso
  - requiere esfuerzo y conocimiento de la implementación
- Incorpora aspectos de dominio, semántica y relaciones entre parámetros
- Un parámetro puede aparecer en múltiples características dificultando su traducción a casos de prueba

Ejemplo

- Considere un programa que toma 2 números enteros y responde si el primero es divisible por el segundo

- Basado en interfaz
  - 2 *inputs* de enteros:
    - mismo dominio para cada uno

  - Posible característica:
    - el signo del número

Ejemplo

- Considere un programa que toma 2 números enteros y responde si el primero es divisible por el segundo.

- Basado en funcionalidad
  - Posibles características:
    - cuando uno es mayor que el otro
    - cuando uno es divisible por el otro
    - cuando uno es distinto al otro

Modelar dominio de entradas: Pasos

1. Identificar funciones *testeables*
   - clases, métodos, sistemas...
2. Buscar parámetros
   - variables, ambiente, DB, estado
3. Modelar dominio de entradas
   - características, particiones y bloques
4. Aplicar un criterio de cobertura
5. Refinar combinaciones de bloques

Criterios de cobertura

- Supongamos las siguientes 3 particiones con sus respectivos bloques:
  - {A, B}
  - {1, 2, 3}
  - {x, y}

ACoC: Cobertura de todas las combinaciones *(All Combinations Coverage)*

- Todas las combinaciones de bloques de todas las características deben ser usadas
- El número de pruebas es el producto del número de bloques en cada característica
  - En el ejemplo anterior 2 x 3 x 2 = 12 casos

(A, 1, x)	(B, 1, x)
(A, 1, y)	(B, 1, y)
(A, 2, x)	(B, 2, x)
(A, 2, y)	(B, 2, y)
(A, 3, x)	(B, 3, x)
(A, 3, y)	(B, 3, y)

- ACoC produce demasiadas pruebas

ECC: Cobertura de cada opción *(Each Choice Coverage)*

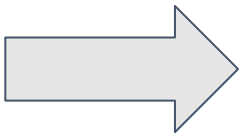
- Se debe utilizar al menos un valor por cada bloque de cada característica en algún caso de prueba.
- La cantidad de pruebas es el número de bloques de la característica “mayor”
  - En el ejemplo anterior anterior existen muchas opciones, tal como {(A, 1, x) ; (B, 2, y) ; (A, 3, x)}

- Es un criterio “débil” al no exigir reglas sobre las combinaciones, por lo que puede ser inefectivo

PWC: Cobertura por Pares *(Pair-wise Coverage)*

- Para cada característica, algún valor de cada bloque debe ser combinado con algún valor de cada bloque de otra característica
- En el ejemplo anterior:

(A, 1)	(B, 1)	(1, x)
(A, 2)	(B, 2)	(1, y)
(A, 3)	(B, 3)	(2, x)
(A, x)	(B, x)	(2, y)
(A, y)	(B, y)	(3, x)
		(3, y)



(A, 1, x)	(B, 1, y)
(A, 2, x)	(B, 2, y)
(A, 3, x)	(B, 3, y)
(A, –, y)	(B, –, x)

TWC: Cobertura por tuplas *(T-wise Coverage)*

- Extensión natural de ***PWC*** donde se utilizan combinaciones de largo ***t*** en lugar de pares.

- Si ***t*** es igual al total de características entonces es idéntico a ***ACoC***

- Esto puede ser costoso y la experiencia sugiere que ***t*** > 2 no es muy útil

BCC: Cobertura de caso base *(Base Choice Coverage)*

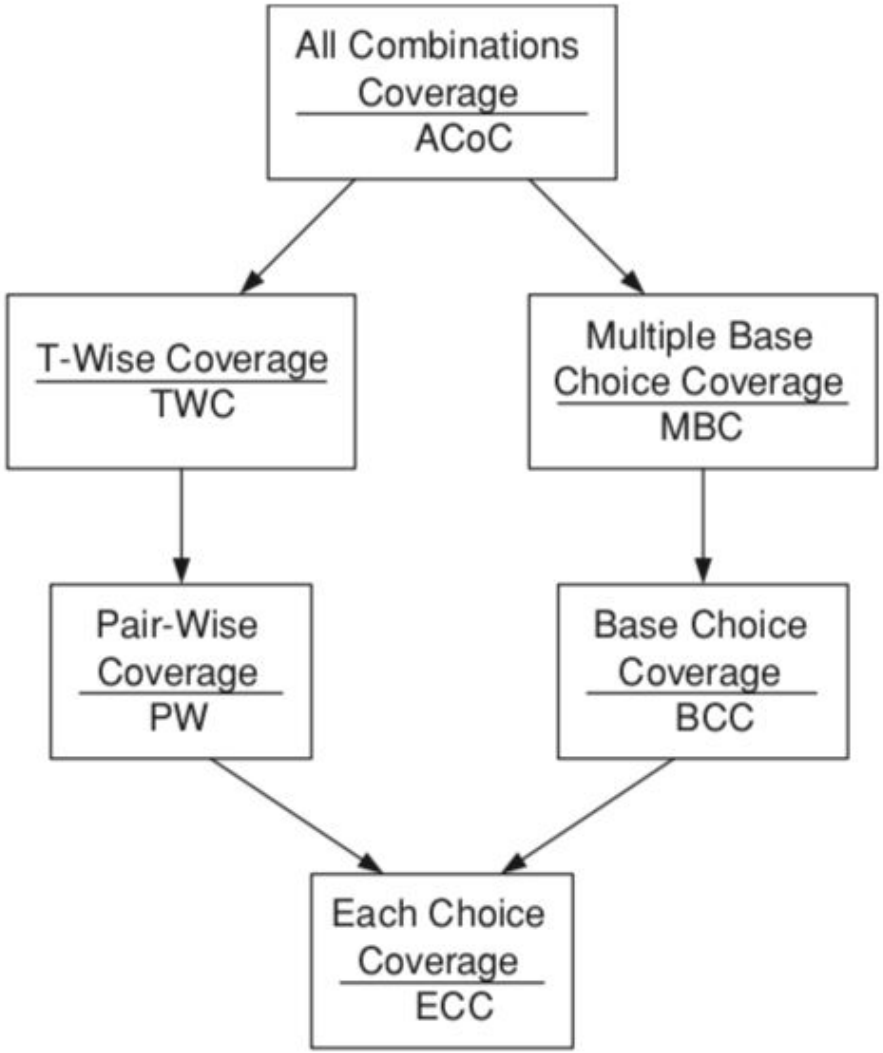
- Se escoge un bloque base por cada característica para formar un *test* base. Luego, se construyen *tests* a partir del *test* base variando una característica a la vez según todos sus bloques.
- En el ejemplo anterior, supongamos (A, 1, x) como base:

(B, 1, x)
(A, 2, x)
(A, 3, x)
(A, 1, y)

MBCC: Cobertura de caso base múltiple *(Multiple Base Choice Coverage)*

- Se identifican múltiples casos base, se buscan combinaciones para cada uno de acuerdo al criterio ***BCC*** y luego se unen todas las combinaciones.

Subsumición



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

Clase 14
Cobertura en base a dominio

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

19 de octubre de 2020