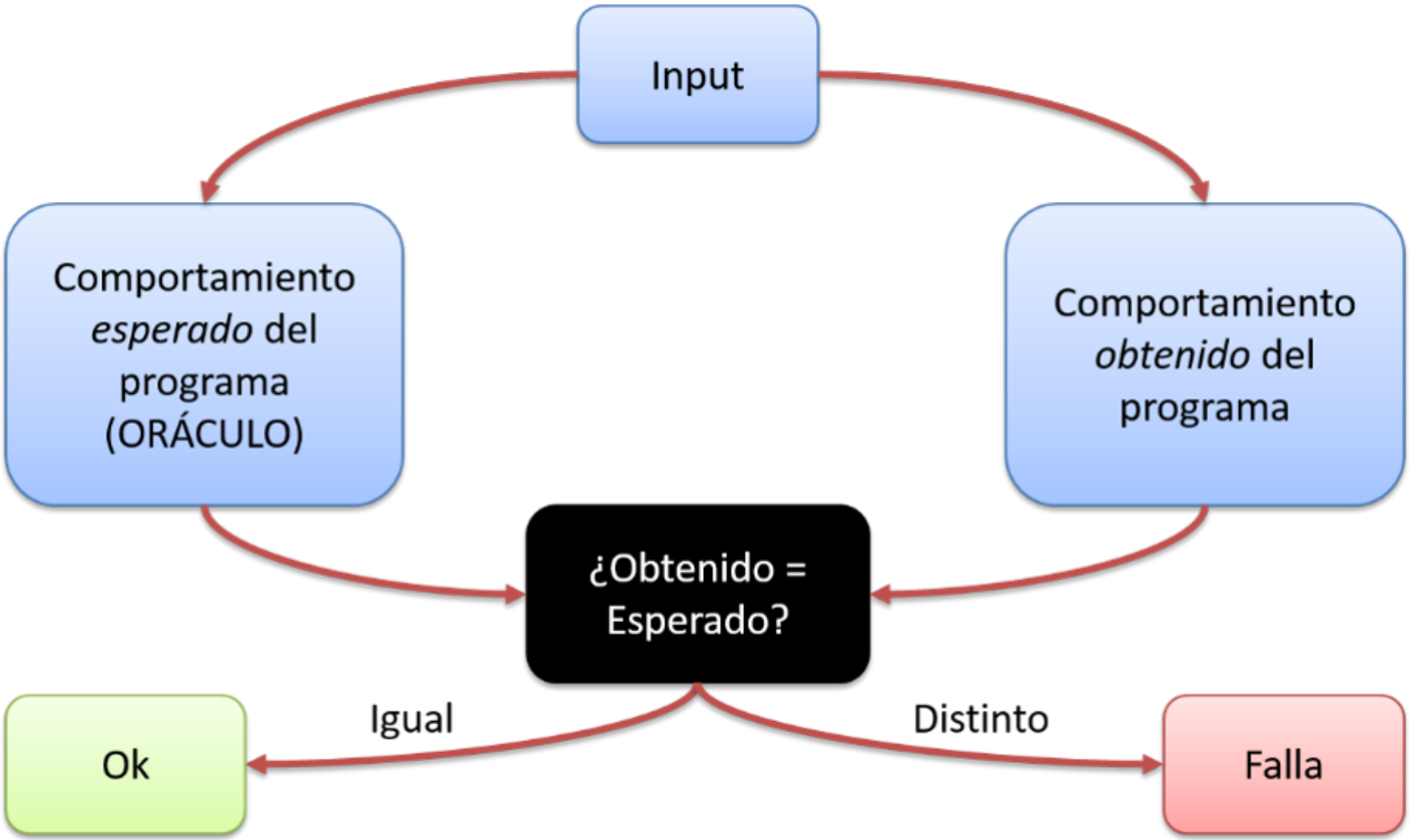
Introducción a la Programación

Algoritmos y Estructuras de Datos I

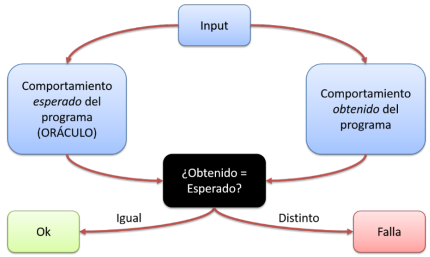
Primer cuatrimestre de 2023

Testing de Caja Negra

¿Cómo se hace testing?



Problemas asociados al testing



- Sobre el input
  - ¿Con qué datos probar?
  - Objetivo: con la menor cantidad de casos de prueba considerar la mayor cantidad de escenarios
- Sobre el resultado esperado
  - ¿Cómo saber qué se espera como resultado esperado?
    - A partir del contrato o especificación
    - A partir del código fuente
- Sobre el resultado obtenido
  - ¿Cómo saber cuál fue el resultado obtenido?
    - A veces puede no ser trivial ver o inspeccionar el resultado obtenido (cuanto más complejo el sistema, más dificil)

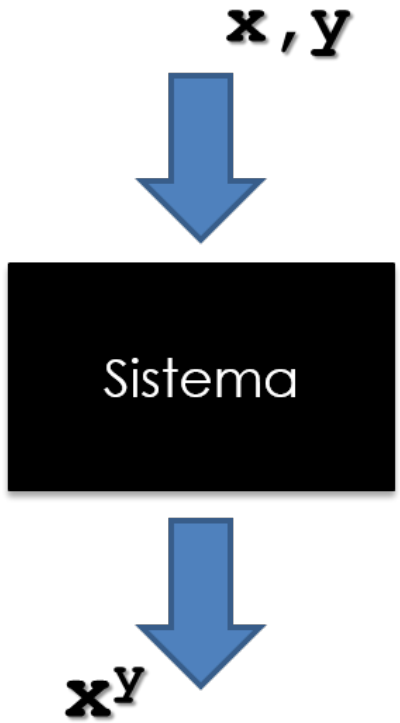
Retomando... ¿Qué casos de test elegir?

1. No hay un algoritmo que proponga casos tales que encuentren todos los errores en cualquier programa.
2. Ninguna técnica puede ser efectiva para detectar todos los errores en un programa arbitrario
3. En ese contexto, veremos dos tipos de criterios para seleccionar datos de test:
   - **Test de Caja Negra**: los casos de test se generan analizando la especificación sin considerar la implementación.
   - **Test de Caja Blanca**: los casos de test se generan analizando la implementación para determinar los casos de test.

Criterios de **caja negra** o funcionales

- Los datos de test se derivan a partir de la descripción del programa sin conocer su implementación.

problema *fastexp*(in $x : \mathbb{Z}$, in $y : \mathbb{Z}$) : $\mathbb{Z}\{$
  requiere: $\{(x \neq 0 \vee y \neq 0) \wedge (y < 0 \rightarrow x \neq 0)\}$
  asegura: $\{res = x^y)\}$
$\}$



Método de Partición de Categorías

- Consiste en una técnica que permite generar casos de prueba de una manera metódica.
- Es aplicable a especificaciones formales, semiformales e inclusive, informales.

El método se puede resumir en los siguientes pasos:

1. Listar todos los problemas que queremos testear
2. Elegir uno en particular
3. Identificar sus parámetros o las relaciones entre ellos que condicionan su comportamiento. Los llamaremos genéricamente factores.
4. Determinar las características relevantes (categorías) de cada factor.
5. Determinar elecciones (choices) para cada característica de cada factor.
6. Clasificar las elecciones: errores, únicos, restricciones, etc
7. Armado de casos, combinando las distintas elecciones determinadas para cada categoría, y detallando el resultado esperado en cada caso.
8. Volver al paso 2 hasta completar todas las unidades funcionales.

Método de Partición de Categorías

Paso 1: Descomponer la solución informática en unidades funcionales

Consiste en enumerar todas las operaciones, funciones, funcionalidades, problemas que se probarán.

problema $esPermutacion(s1, s2 : seq\langle T \rangle) : Bool$ {
  asegura: $\{res = true \leftrightarrow ((\forall e : T)(cantidadDeApariciones(s1, e) = cantidadDeApariciones(s2, e)))\}$
}

problema $cantidadDeApariciones(s : seq\langle T \rangle, e : T) : \mathbb{Z}$ {
  requiere: $\{e \in s\}$
  requiere: $\{|s| > 0\}$
  asegura: $\{res = \sum_{i=0}^{|s|-1}(\text{if } s[i] = e \text{ then } 1 \text{ else } 0 \text{ fi})\}$
}

En este caso:

- esPermutacion
- cantidadDeApariciones

Método de Partición de Categorías

Paso 2: Elegir una unidad funcional

Lo ideal es llegar a testear todas las unidades funcionales. Un buen criterio, es empezar por aquellas que *son utilizadas* por otras.

En este caso:

- esPermutacion
- cantidadDeApariciones

Método de Partición de Categorías

Paso 3: Identificar factores

Esto pueden ser los parámetros del problema a testear (si el sistema es más complejo... podrían ser otros factores).

problema $cantidadDeApariciones(s : seq\langle T \rangle, e : T) : \mathbb{Z}$ {
  requiere: $\{e \in s\}$
  requiere: $\{|s| > 0\}$
  asegura: $\{res = \sum_{i=0}^{|s|-1} (\text{if } s[i] = e \text{ then } 1 \text{ else } 0 \text{ fi})\}$
}

En este caso:

- Tomamos T como Z(porque vamos a buscar datos concretos para probar)
- s: $seq\langle \mathbb{Z} \rangle$
- e: Z

Método de Partición de Categorías

Paso 4: Determinar categorías

Las categorías son distintas características de cada factor, o características que relacionan diferentes factores, y que tienen influencia en los resultados. Son el resultado del análisis de toda la información disponible sobre la funcionalidad a testear.

En nuestro ejemplo, para cada parámetro podemos determinar las siguientes caracteristicas:

- s: $seq\langle\mathbb{Z}\rangle$
  - ¿Tiene elementos?
- e: Z
  - En este caso, para e no se distingue ninguna característica interesante
- Relación entre s y e (esta relación puede ser interesante)
  - ¿Pertenece e a s?

Método de Partición de Categorías

Paso 5: Determinar elecciones

Se trata de buscar los conjuntos de valores donde se espera un comportamiento similar. Se basa en las especificaciones, la experiencia, el conocimiento de errores.

En nuestro ejemplo, para cada categoría, determinamos sus elecciones o choices:

- s: $seq\langle\mathbb{Z}\rangle$
  - ¿Tiene elementos?
    - Si
    - No
- e: Z
- Relación entre s y e
  - ¿Pertenece e a s?
    - Si
    - No

Método de Partición de Categorías

Paso 6: Clasificar las elecciones

Se trata de identificar algunas propiedades o restricciones de las elecciones en el marco de la unidad funcional.
Las clasificaciones más comunes son:

- Error: Se clasificarán como error aquellas elecciones que por si mismas determinen que como resultado de la ejecución el sistema debe detectar un error o que no está definido su comportamiento.
- Otros posibles valores: Único, Restricción, etc. (más adelante ampliaremos).

En nuestro ejemplo, para cada elecciones o choices, analizamos si debemos clasificarlo especialmente:

- s: $seq\langle\mathbb{Z}\rangle$
  - ¿Tiene elementos?
    - Si
    - No [ERROR]
- e: Z
- Relación entre s y e
  - ¿Pertenece e a s?
    - Si
    - No [ERROR]

Método de Partición de Categorías

Paso 7: Armar los casos de test

Finalmente, se combinarán las distintas elecciones de las categorías consideradas generando los distintos casos de test. Cada combinación es un caso de test, el cual deberá tener identificado con claridad su resultado esperado.
En nuestro ejemplo, deberemos combinar:

- ¿s tiene elementos?: Si
- ¿s tiene elementos?: No
- ¿e pertenece a s?: Si
- ¿e pertenece a s?: No

Tenemos 4 elecciones posibles a combinar

- ¿Nos interesan todas las combinaciones?
- ¿Cuántos casos de test tenemos?

Método de Partición de Categorías

Paso 7: Armar los casos de test

Finalmente, se combinarán las distintas elecciones de las categorías consideradas generando los distintos casos de test. Cada combinación es un caso de test, el cual deberá tener identificado con claridad su resultado esperado.
En nuestro ejemplo, deberemos combinar:

- ¿s tiene elementos?: Si
- ¿s tiene elementos?: No
- ¿e pertenece a s?: Si
- ¿e pertenece a s?: No

Tenemos 4 elecciones posibles a combinar

- ¿Nos interesan todas las combinaciones?
- ¿Cuántos casos de test tenemos?
- Las elecciones marcadas como ERROR y UNICO, suelen no ser combinables (recortando así la cantidad de combinaciones a realizar).
- Las elecciones RESTRICCION, condicionan las combinaciones posibles.
- Tip: estas elecciones son las primeras a considerar en el armado de casos.

Método de Partición de Categorías

Paso 7: Armar los casos de test

En este caso, sólo nos quedan 3 casos interesantes (nos ahorramos 1)

- Por cada caso, debemos describir su resultado esperado: es importante indicar si el resultado será un posible resultado correcto u esperable o un error o comportamiento indefinido.
- Recordar que los casos de prueba definido serán una herramienta que eventualmente otra persona pueda ejecutar los test: eligiendo datos concretos y comparando el resultado obtenido con el esperado.

Característica	¿s tiene elementos?	¿e pertenece a s?	Resultado esperado	Comentario
Caso 1: S sin elementos	No	-	ERROR: no está espeficado que sucede en este caso.	Como la elección No es un ERROR, no importa el valor
Caso 2: E no pertenece	-	No	ERROR: no está espeficado que sucede en este caso.	Como la elección No es un ERROR, no importa el valor
Caso 3: S tiene elementos y e Pertenece	Si	Si	OK: el resultado obtenido debe ser igual a la cantidad de	

- La tabla es una representación gráfica y práctica de los casos.
- Suele ocurrir, que las primeras columnas son siempre de aquellas elecciones que tienen errores, unicos o restricciones entre sus posibles valores: porque descartan casos hacia la derecha.

Método de Partición de Categorías

Paso 6: Clasificar las elecciones - Volviendo un paso atrás

Se trata de identificar algunas propiedades o restricciones de las elecciones en el marco de la unidad funcional.
Las clasificaciones más comunes son:

- Único: Son aquellas elecciones que no necesitan combinarse con ninguna otra elección para determinar el resultado esperado.

problema *siElPrimeroEsCinco*$(x : \mathbb{Z}, y : \mathbb{Z}) : \mathbb{Z}$ {
  asegura: $\{x = 5 \longrightarrow res = 5\}$
  asegura: $\{x \neq 5 \longrightarrow res = x + y$
}

- Factor: x
  - Característica: valor
    - Elección: Igual a 5 [UNICO]
    - Elección: Distinto que 5
- Factor: y

Método de Partición de Categorías

Paso 6: Clasificar las elecciones - Volviendo un paso atrás

Se trata de identificar algunas propiedades o restricciones de las elecciones en el marco de la unidad funcional.
Las clasificaciones más comunes son:

- Único: Son aquellas elecciones que no necesitan combinarse con ninguna otra elección para determinar el resultado esperado.

problema $siElPrimeroEsCincoHaceOtraCosa(x : \mathbb{Z}, y : \mathbb{Z}, z :) : \mathbb{Z}$ {
  requiere: $\{x = 5 \longrightarrow z \neq 0\}$
  asegura: $\{x = 5 \longrightarrow res = x + y/z\}$
  asegura: $\{x \neq 5 \longrightarrow res = x + y$
}

- Factor: x
  - Característica: valor
    - Elección: Igual a 5 [RESTRICCION]
    - Elección: Distinto que 5
- Factor: y
- Factor: z
  - Característica: ¿Es distinto de 0?
    - Elección: Si
    - Elección: No Sólo si x=5 x RESTRICCION

Método de Partición de Categorías

Si hay otra funcionalidad a testear $\longrightarrow$Paso 2: Elegir una unidad funcional

Si probamos esPermutacion luego de haber testeado cantidadDeApariciones, podemos asumir que cantidadDeApariciones funciona correctamente.

En este caso:

- esPermutacion
- cantidadDeApariciones

Y repetimos el proceso

Método de Partición de Categorías

Comentarios finales

- Este es sólo un método más (de otros tantos que existen) para encarar el problema de generar casos de prueba.
- No se evaluará su uso de manera rigurosa en la materia
  - La intención es que cuenten con alguna herramienta en caso de que se encuentren frente a la situación de no saber cómo testear sus problemas
  - No existe una única forma de generar casos de prueba!
  - Lo importante, es que sus casos de prueba abarquen, en la medida de lo posible, todas las casuísticas posibles con respecto a loss parámetros de entrada.