Ingeniería del Software II

Parcial #1 – Generación Automática de Tests

Ej1	Ej2	Ej3	Ej4	Nota
B	B	B	B	

El exámen es a **libro abierto**. Cada ejercicio se evaluará como **Bien, Regular** o **Mal.** Para aprobar el examen es necesario tener al menos 2 ejercicios Bien y al menos 1 ejercicio Regular.

Bien = 2,5p, Regular = 1,25p, Mal = 0p.

Ejercicio 1

Sea el siguiente programa:

```
1  def test_me(k: int, j: int) -> int:
2    i:int=0
3    r:int=0
4    while i<2: # C1
5      if k==j: # C2
6        r=r+1
7      i=i+1
8    return r
```

Exhibir usando el operador de mutación ABS[1]:

a. Un mutante con un test que lo detecte. Indicar cuál es el test case que lo detecta.

b. Un mutante equivalente tal que no exista ningún test case que lo detecte.

En ambos casos indicar la línea que se modificó.

Ejercicio 2

Sea el programa `test_me` del ejercicio #1.

a. Completar la siguiente tabla con la ejecución simbólica dinámica del programa de forma manual, indicando para cada iteración:

- El input concreto utilizado
- La condición de ruta (i.e. "path condition") que se produce al ejecutar el input concreto, asumiendo que el valor simbólico inicial es $k = k_0$ y $j = j_0$.
- La fórmula lógica (no es necesario escribirla en SMTLib) que se envía al demostrador de teoremas de acuerdo al algoritmo de ejecución simbólica dinámica.
- El resultado posible que podría producir un demostrador de teoremas (ej: Z3).

b. Describir el árbol de cómputo del programa explorado durante la ejecución simbólica dinámica del programa

Iteración	Input Concreto	Condición de Ruta	Fórmula enviada al demostrador	Resultado posible
1	k=0, j=0	...	...	...
2	...	...	...	...
...	...	...	...	...

[1] modifica una expresión artimética E reemplazándola por $abs(E)$, $-abs(E)$, o la constante 0

Ejercicio 3

Sea el programa `test_me` del ejercicio #1:

a. ¿Existe algún test suite tal que se ejecute normalmente y se obtenga un valor de retorno de la función `test_me` donde la *branch distance* a la decisión false del `while` sea mayor que 0?

b. Sea el siguiente test suite:

```
class TestSuite(unittest.TestCase):
  def test_1(self):
    self.assertEqual(0, test_me(1,0))
```

I. ¿Cuál es el valor de la distancia de branch no normalizada para cada decisión si ejecutamos el test suite?

branch	distancia true	distancia false
4: `while i<2:`		
5: `if k==j:`		

II. ¿Cuál es el cubrimiento de branches que logra el test suite?

Ejercicio 4

Sea el siguiente programa:

```
1  def parse_email(email: str) -> List[Optional[str]]:
2      parts = email.split('@')
3      if len(parts) != 2:
4          return [None, None, None]
5      else:
6          local_part, domain = parts
7          if not local_part:
8              return [None, None, None]
9          else:
10             if not domain or '.' not in domain:
11                 return [None, None, None]
12             else:
13                 last_dot_index = domain.rfind('.')
14                 domain_part = domain[:last_dot_index]
15                 top_level_domain = domain[last_dot_index + 1:]
16                 return [local_part, domain_part, top_level_domain]
```

Asumiendo que tenemos un boosted greybox fuzzer con exponente a=3. Sea el siguiente conjunto inicial de inputs:

- #1: ``user@example.com''
- #2 ``john.doe@example.co.uk''
- #3 ``alice.smith@subdomain.example.org''
- #4 ``info@company.com''
- #5 ``support@sub.subdomain.example.net''
- #6``invalid-email''
- #7 ``missing_domain@''
- #8 ``@missing_local_part.com''
- #9 ``missing_local_part_and_domain''
- #10 ``user@[IPv6:2001:db8::1]''
- #11``user+tag@example.com''
- #12 ``john.smith@subdomain.example.org''

a. Por cada input en el conjunto inicial, indicar su camino de ejecución, y su energía asignada luego de ejecutar todo el conjunto inicial.

b. ¿Cuál es la probabilidad que el fuzzer elija el input ``alice.smith@subdomain.example.org" para mutar?

(1)

a) PROPONGO UN MUTANTE CON UN CAMBIO EN LA LÍNEA 6, REEMPLAZANDOLA POR LA SIGUIENTE:

r = 0 ✓ (ORIGINALMENTE r = r + 1)

ASÍ, REEMPLAZO LA EXPRESIÓN ARITMÉTICA r+1 POR 0.

ADEMÁS, PROPONGO EL SIGUIENTE TEST:

TEST-1:

ASSERT_EQUALS(TEST_ME(0,0), 2) ✓

ESTE TEST DEBERÍA FALLAR PARA EL MUTANTE, YA QUE HACER TEST_ME(0,0) EN EL CÓDIGO MUTANTE DEBERÍA DEVOLVER 0, PORQUE NO INCREMENTA NUNCA LA VARIABLE r. ✓

b) PODEMOS CONSEGUIR UN MUTANTE EQUIVALENTE MODIFICANDO, NUEVAMENTE, LA LÍNEA 6, PERO AHORA DE LA SIGUIENTE MANERA:

r = ABS(r + 1) ✓

DE ESTA MANERA CONSEGUIMOS UN MUTANTE EQUIVALENTE, YA QUE EN EL CÓDIGO ORIGINAL r ES INICIALIZADA EN 0 E INCREMENTADA EN 1. POR ESTO r ES SIEMPRE MAYOR O IGUAL A 0, Y HACER r+1 TAMBIÉN LO SERÁ.

CÓDIGO DEL MUTANTE DEL (a)

```
DEF TEST_ME(k:INT, j:INT) → INT:
    i: INT = 0
    r: INT = 0
    WHILE i < 2:
        IF k == j:
            r = 0
        i = i + 1
    RETURN r
```

DEL (b)

```
DEF TEST_ME(k:INT, j:INT) → INT:
    i: INT = 0
    r: INT = 0
    WHILE i < 2:
        if k == j
            r = ABS(r + 1)
        i = i + 1
    RETURN r.
```

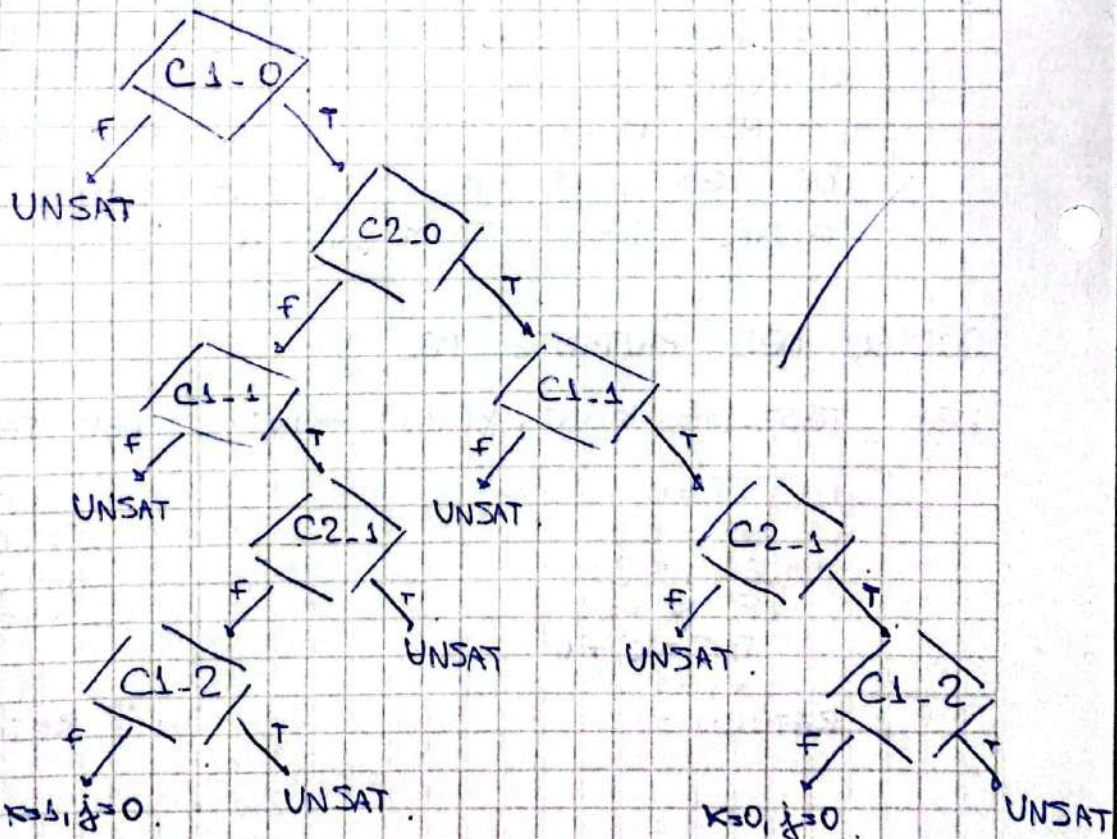
✓

(2)

a)

IT	INPUT CONCRETO	CONDICIÓN DE RUTA	FÓRMULA	RESULTADO
1	$k=0,\ j=0$	$0<2 \ \&\ k_0=j_0 \ \&\ 1<2 \ \&\ k_0=j_0 \ \&\ !2<2$	$0<2 \ \&\ k_0=j_0 \ \&\ 1<2 \ \&\ k_0=j_0 \ \&\ 2<2$	UNSAT
			$0<2 \ \&\ k_0=j_0 \ \&\ 1<2 \ \&\ !k_0=j_0$	UNSAT
			$0<2 \ \&\ k_0=j_0 \ \&\ !1<2$	UNSAT
			$0<2 \ \&\ !k_0=j_0$	$k_0=1,\ j_0=0$
2	$k=1,\ j=0$	$0<2 \ \&\ !k_0=j_0 \ \&\ 1<2 \ \&\ !k_0=j_0 \ \&\ !2<2$	$0<2 \ \&\ !k_0=j_0 \ \&\ 1<2 \ \&\ !k_0=j_0 \ \&\ 2<2$	UNSAT
			$0<2 \ \&\ !k_0=j_0 \ \&\ 1<2 \ \&\ k_0=j_0$	UNSAT
			$0<2 \ \&\ !k_0=j_0 \ \&\ !1<2$	UNSAT
			$!0<2$	UNSAT

b)



③

a) NO, NO EXISTE. SI ESE FUERA EL CASO, ENTONCES LO QUE QUERRÍA DECIR ES QUE EL CICLO NUNCA TERMINA.
SIN EMBARGO, EL CICLO ESTÁ BIEN DEFINIDO CON UNA CONDICIÓN DE PARADA (i<2) Y CON LA VARIABLE i SIEMPRE INCREMENTADO EN C/ ITERACIÓN.
EN PARTICULAR, EL CICLO SIEMPRE ITERARA 2 VECES Y PARARA.

b)

I

BRANCH	DISTANCE TRUE	DISTANCE FALSE
WHILE i<2	0	0
IF K==j	1	0

COMO EXPLIQUE ANTES, EL CICLO SE EJECUTA 2 VECES, POR LO QUE TANTO PARA TRUE Y FALSE LA DISTANCIA SERÁ CERO.
EN EL CASO DEL IF, DADO QUE K Y j NO SON IGUALES, LA CONDICIÓN NO SE CUMPLIRÁ, POR ESO DA 0 LA DISTANCE A FALSE.
POR EL MISMO MOTIVO, LA DISTANCE A TRUE NO ES CERO. EN PARTICULAR ES ABS(K-j) = 1.

II COMO SE PUEDE VER ARRIBA EN LA TABLA, 3 BRANCHES SON CUBIERTAS (DISTANCE =0) Y 1 NO (DISTANCE DISTINTA A CERO)

BRANCH_COVERAGE = 3/4.

④

a)

INPUT	CAMINO	ENERGIA
#1	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$
#2	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$
#3	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$
#4	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$
#5	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$

NOTA (SIGO ATRÁS).

INPUT	CAMINO	ENERGÍA
#6	1,2,3,4	$1/2^3$
#7	1,2,3,5,6,7,9,10,11	$1/2^3$
#8	1,2,3,5,6,7,8	$1/1^3 = 1$
#9	1,2,3,4	$1/2^3$
#10	1,2,3,5,6,7,9,10,11	$1/2^3$
#11	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$
#12	1,2,3,5,6,7,9,10,12,13,14,15,16	$1/7^3$

b) LA PROBABILIDAD ES:

$$\frac{1/7^3}{1/7^3 \times 7 + 1/2^3 \times 2 + 1/2^3 \times 2 + 1}$$