Taller #3 - DSE con Z3

Grupo F (#6)

Nombre	Mail	LU
Manuel Panichelli	panicmanu@gmail.com	72/18
Elias Cerdeira	eliascerdeira@gmail.com	692/12

Ejercicio 1

- a. `¬(x ∨ y) ≡ (¬x ∧ ¬y)`

  Resultado: sat

  Archivo `especificaciones/ejercicio1a.smt`

- b. `(x ∧ y) ≡ ¬(¬x ∨ ¬y)`

  Resultado: sat

  Archivo `especificaciones/ejercicio1b.smt`

- c. `¬(x ∧ y) ≡ ¬(¬x ∧ ¬y)`

  Resultado: sat

  Archivo `especificaciones/ejercicio1c.smt`

Ejercicio 2

- a. `3x + 2y = 36`

  Archivo `especificaciones/ejercicio2a.smt`

  Resultado:

  ```
  sat

  (model
    (define-fun y () Int
      0)
    (define-fun x () Int
      12)
  )
  ```

- b. `5x + 4y = 64`

  Archivo `especificaciones/ejercicio2b.smt`

Resultado:

```
sat

(model
  (define-fun y () Int
    1)
  (define-fun x () Int
    12)
)
```

- c. `x * y = 64`

  Archivo `especificaciones/ejercicio2c.smt`

  ```
  sat

  (model
    (define-fun y () Int
      1)
    (define-fun x () Int
      64)
  )
  ```

Ejercicio 3

Resultado:

```
sat
(model
  (define-fun a2 () Real
    4.0)
  (define-fun a1 () Real
    0.0)
  (define-fun a3 () Real
    1.0)
)
```

Ejercicio 4

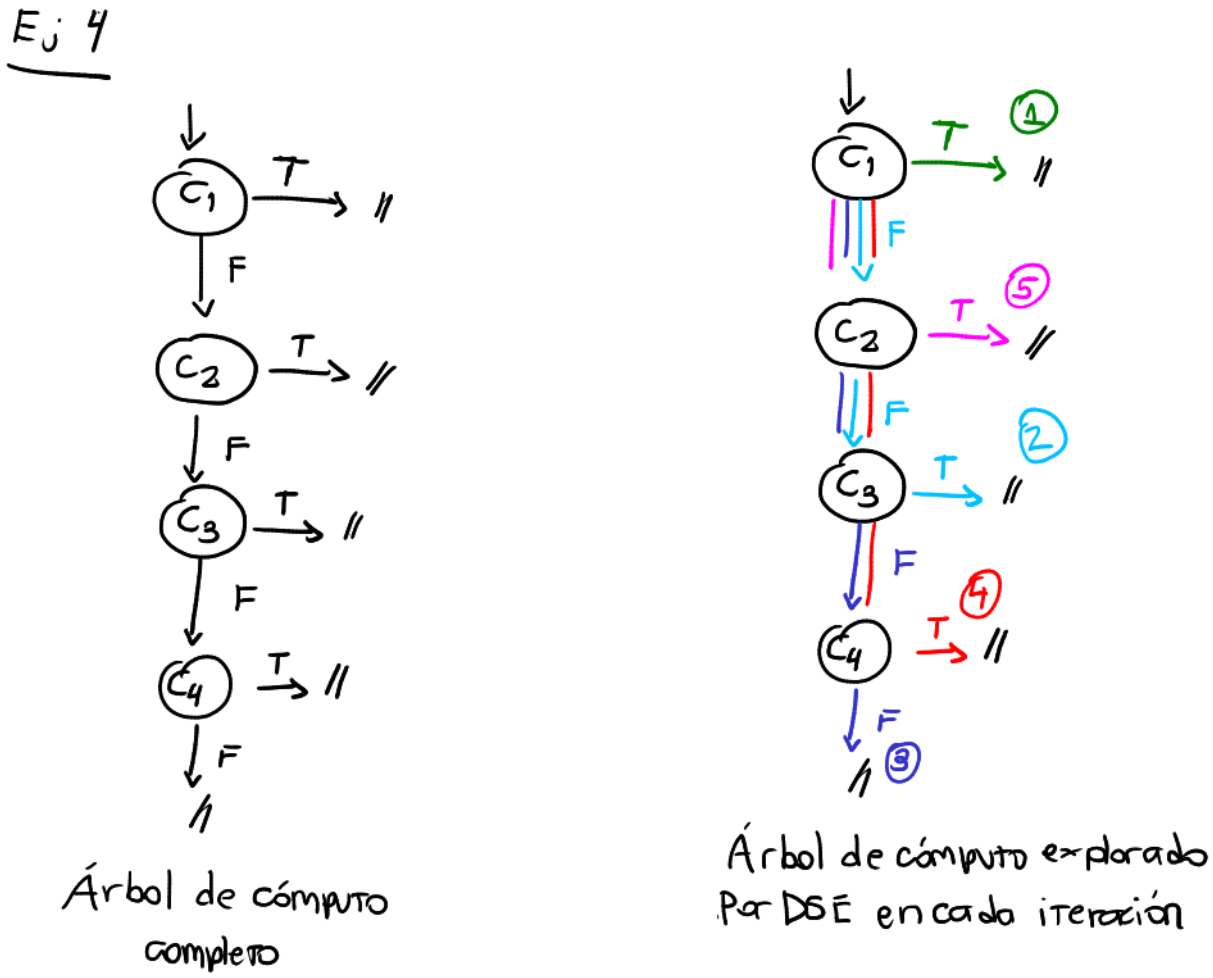
- a.

  ```
  # Path conditions
  c1: a_0 <= 0 || b_0 <= 0 || c_0 <= 0
  c2: !( a_0 + b_0 > c_0 && a_0 + c_0 > b_0 && b_0 + c_0 > a_0 )
  c3: a_0 == b_0 && b_0 == c_0
  ```

```
c4: a_0 == b_0 || b_0 == c_0 || a_0 == c_0

# Z3
(assert (= c1 (or
    (<= a 0)
    (or (<= b 0) (<= c 0)))))

(assert (= c2 (not (and (> (+ a b) c) (and (> (+ a c) b) (> (+ b c)
a))))))

(assert (=
    c3
    (and
        (= a b)
        (= b c))))

(assert (= c4 (or
    (= a b)
    (or (= b c) (= a c)))))

z_it1: (assert (not c1))

z_it2: (assert
    (and
        (not c1)
        (and
            (not c2)
            (not c3))))

z_it3: (assert (and
    (not c1)
    (and
        (not c2)
        (and
            (not c3)
            c4))))

z_it4: (assert (and (not c1) c2)
```

It	Input	Route cond	Spec Z3	Res Z3
1	a=0, b=0, c=0	c1	z_it1	a=1, b=1, c=1
2	a=1, b=1, c=1	!c1 ^ !c2 ^ c3	z_it2	a=2, b=3, c=4
3	a=1, b=2, c=3	!c1 ^ !c2 ^ !c3 ^ !c4	z_it3	a=2, b=2, c=1
4	a=2, b=2, c=1	!c1 ^ !c2 ^ !c3 ^ c4	z_it4	a=1, b=1, c=2
5	a=1, b=1. c=2	!c1 ^ c2	END	END

- b. 100%
- c.



Ejercicio 5

- a.

```
# Path conditions
l0: 0 < 3
l1: 1 < 3
l2: 2 < 3
l3: 3 < 3
c0: array[0] + k == 0
c1: array[1] + k == 0
c2: array[2] + k == 0

# Z3
(assert (= c0 (= 0 (+ a0 k))))
(assert (= c1 (= 0 (+ a1 k))))
(assert (= c2 (= 0 (+ a2 k))))

z_it1: (assert (and (not c0) (and (not c1) c2)))
z_it2: (assert (and (not c0) c1))
z_it3-1: (assert (and (not c0) (and c1 c2)))
z_it3-2: (assert c0)
z_it4-1: (assert (and c0 (and (not c1) c2)))
z_it4-2: (assert (and c0 c1))
```

It	Input	Route cond	Spec Z3	Res Z3
1	k=0.0	l0 ^ !c0 ^ l1 ^ !c1 ^ l2 ^ !c2 ^ !l3	z_it1	k=-3.0
2	k=-3.0	l0 ^ !c0 ^ l1 ^ !c1 ^ l2 ^ c2 ^ !l3	z_it2	k=-1.0
3	k=-1.0	l0 ^ !c0 ^ l1 ^ c1 ^ l2 ^ !c2 ^ !l3	z_it3-1	unsat
			z_it3-2	k=-5
4	k=-5.0	l0 ^ c0 ^ l1 ^ !c1 ^ l2 ^ !c2 ^ !l3	z_it4-1	unsat
			z_it4-2	unsat
			END	END

- b. Tiene 100% de branch coverage porque se calcula sobre el CFG y se evalúan ambas condiciones (la condición del loop y el if dentro) por true y false.
- c.