Taller #4 | DSE

Ingeniería del Software 2

2do Cuatrimestre 2020

Menú del día

- ¿En dónde estamos?
- Repaso DSE
- Consideraciones y enunciado

Menú del día

- ***¿En dónde estamos?***
- Repaso DSE
- Consideraciones y enunciado

¿En dónde estamos? | *Contexto*

Tres secciones generales

Análisis Estático de Programas

Testing Automatizado de Software

Verificación de Programas Concurrentes

¿En dónde estamos? | *Testing Automatizado*

Primera Parte

Introducción al Testing Automatizado | ***Nociones Básicas***

Random Testing | ***Técnicas Naive***

Generación Automática de Tests | ***Randoop / Korat***

¿En dónde estamos? | *Testing Automatizado*

Segunda Parte

Ejecución Simbólica Dinámica (DSE)

Search Based Testing

Menú del día

- ¿En dónde estamos?
- ***Repaso DSE***
- Consideraciones y enunciado

DSE | *Idea*

1. Ejecutar el SUT con algún input semilla.

Sujeto Bajo Prueba

```
void test_me(int x, int y) {
    int z = x * 2;
    if (z == x)
        if (x > y + 10)
            ERROR;
}
```

DSE | *Idea*

1. Ejecutar el SUT con algún input semilla.
2. Recolectar constraints a lo largo de la ejecución. Por cada branch recorrido, se agrega una constraint.

Sujeto Bajo Prueba

```
void test_me(int x, int y) {
    int z = x * 2;
    if (z == x)
        if (x > y + 10)
            ERROR;
}
```

DSE | *Idea*

1. Ejecutar el SUT con algún input semilla.
2. Recolectar constraints a lo largo de la ejecución. Por cada branch recorrido, se agrega una constraint.
3. Modificar la path condition para predicar sobre un nuevo camino no explorado.

Sujeto Bajo Prueba

```
void test_me(int x, int y) {
    int z = x * 2;
    if (z == x)
        if (x > y + 10)
            ERROR;
}
```

DSE | *Idea*

1. Ejecutar el SUT con algún input semilla.
2. Recolectar constraints a lo largo de la ejecución. Por cada branch recorrido, se agrega una constraint.
3. Modificar la path condition para predicar sobre un nuevo camino no explorado.
4. Usar un demostrador de teoremas para obtener un nuevo input en base a la path condition modificada.

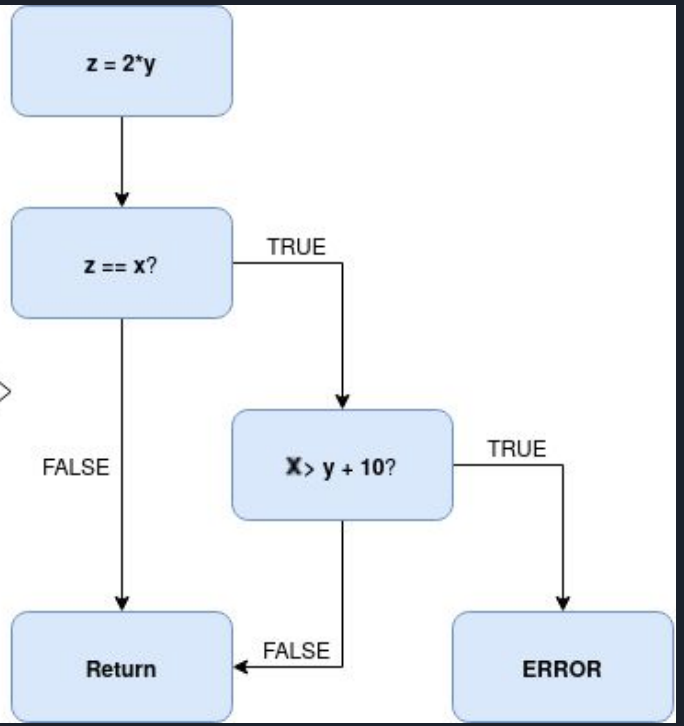
Sujeto Bajo Prueba

```
void test_me(int x, int y) {
    int z = x * 2;
    if (z == x)
        if (x > y + 10)
            ERROR;
}
```

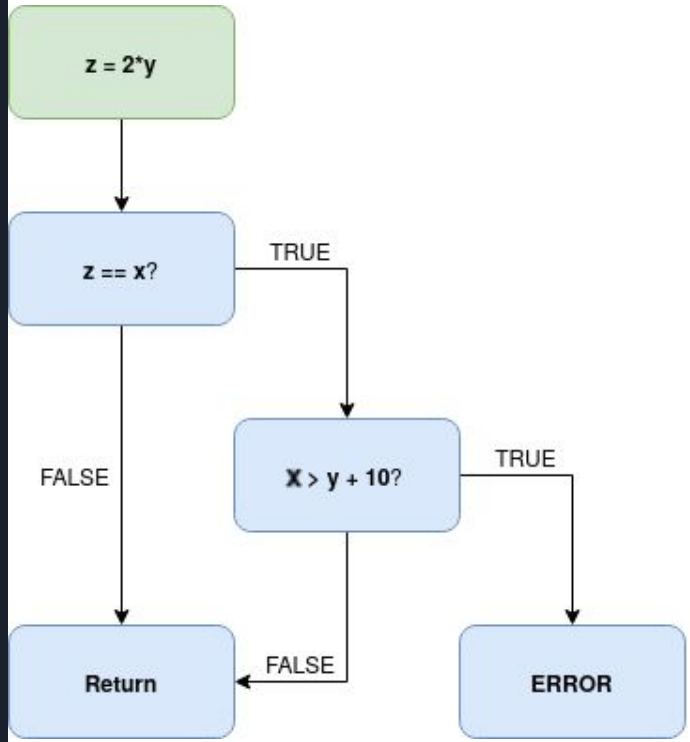
DSE | *Paso 1: Recolección Simbólica*

Representación en el grafo de control de flujo

```
Void test_me(int x, int y) {
        int z = x*2;
        if (z == x)
                if (x > y + 10)
                        ERROR;
}
```

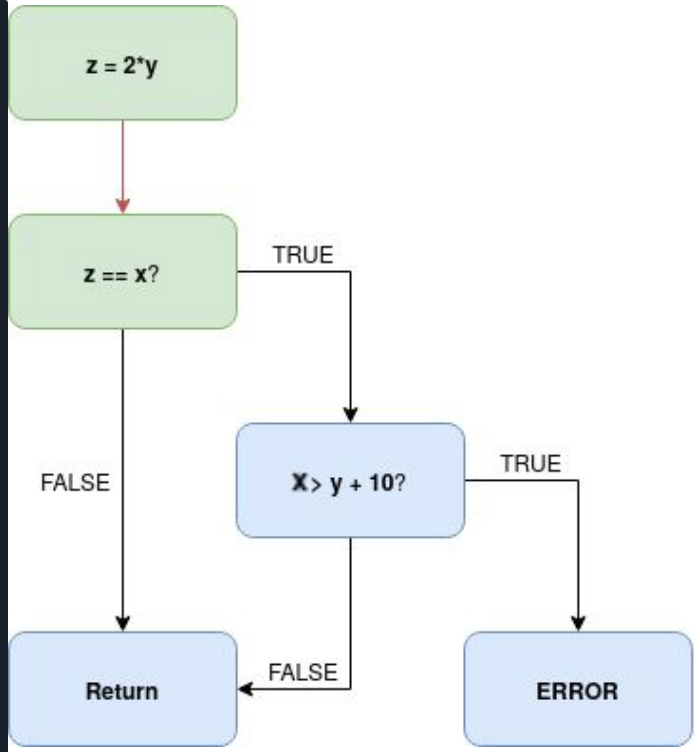


DSE | *Paso 1: Recolección Simbólica*



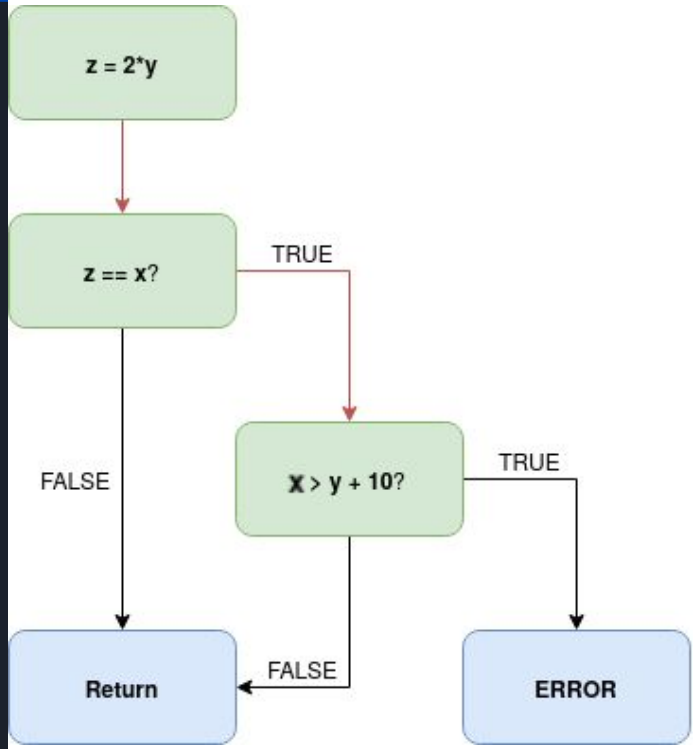
Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2	x = x_0	
y = 1	y = y_0	

DSE | *Paso 1: Recolección Simbólica*



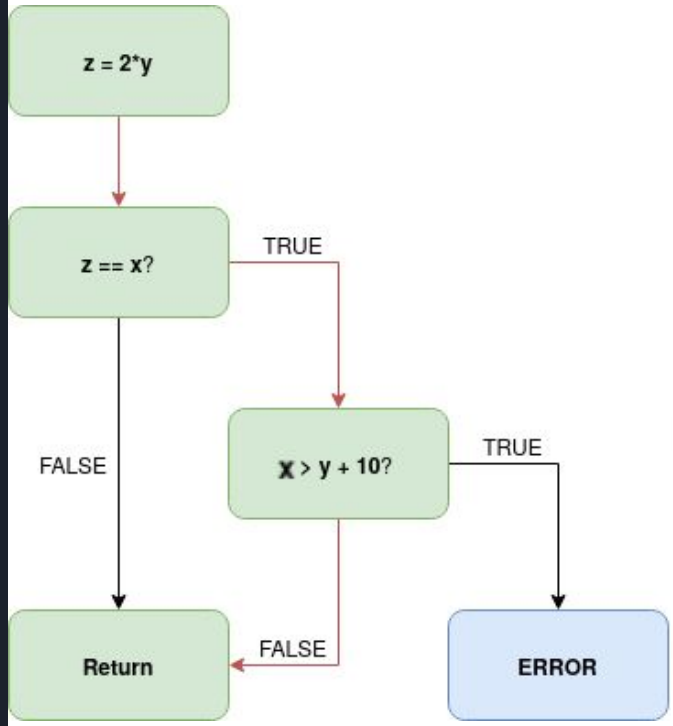
Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2	x = x_0	
y = 1	y = y_0	
z = 2	z = 2*y_0	

DSE | *Paso 1: Recolección Simbólica*



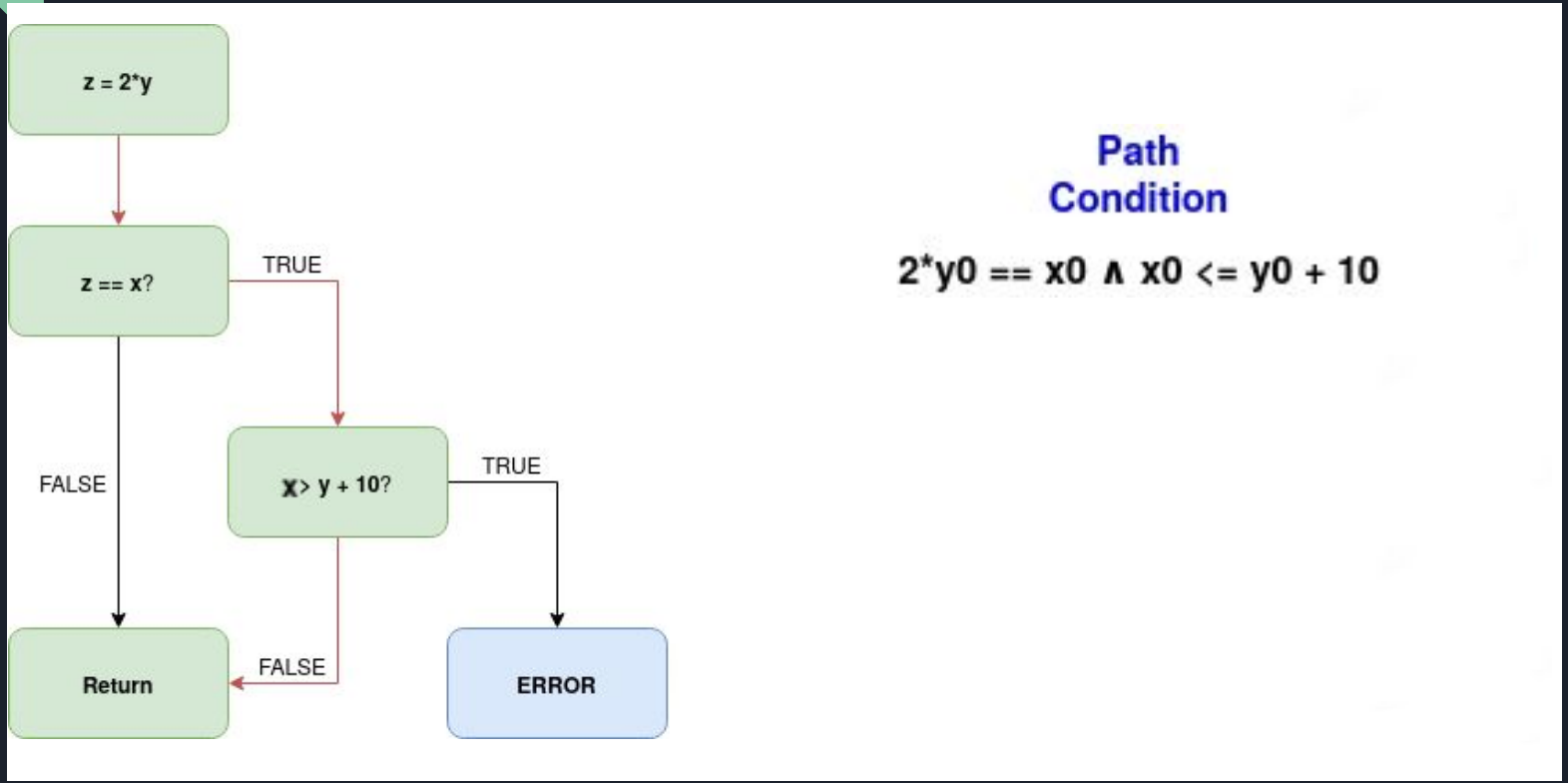
Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2	x = x_0	$2*y_0 == x_0$
y = 1	y = y_0	
z = 2	z = $2*y_0$	

DSE | *Paso 1: Recolección Simbólica*

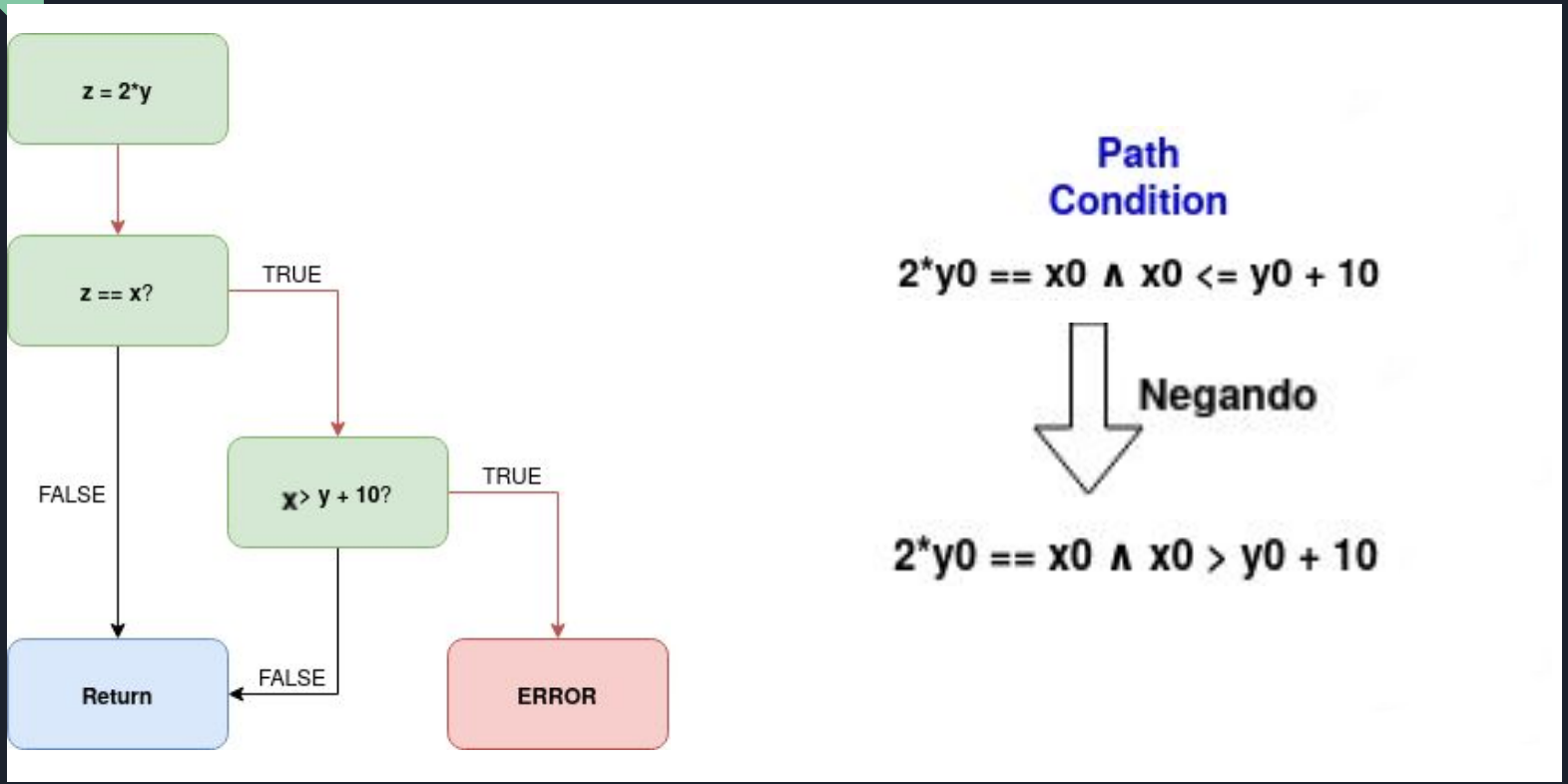


Concrete Execution		Symbolic Execution
concrete state	symbolic state	path condition
x = 2	x = x_0	$2*y_0 == x_0$
y = 1	y = y_0	
z = 2	z = $2*y_0$	$x_0 <= y_0+10$

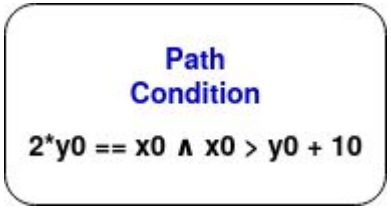
DSE | *Paso 2: Nueva path condition*



DSE | *Paso 2: Nueva path condition*

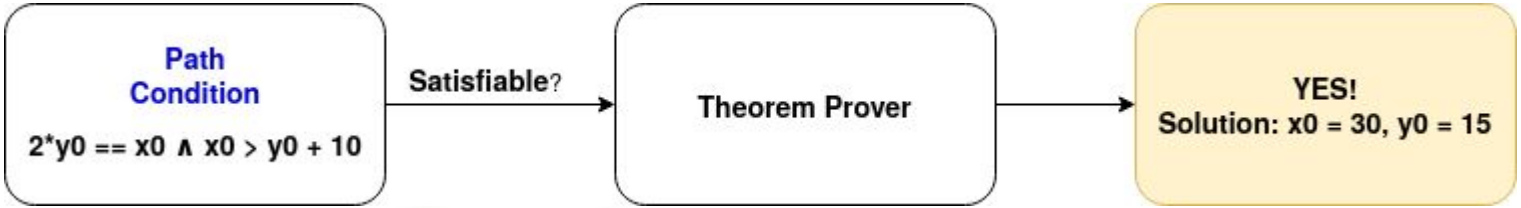


DSE | *Paso 3: Generación de un nuevo input*



DSE | *Paso 3: Generación de un nuevo input*

DSE | *Paso 3: Generación de un nuevo input*



DSE | *¿En qué se enfoca este taller?*

- Ejecución de la técnica y el contraste con Randoop
  - ¿Cuándo conviene usar DSE?
- Resolución de una path condition por medio de un demostrador de teoremas
  - Modelar la path condition en un demostrador de teoremas
  - Vamos a usar Z3 con un tutorial en el enunciado

Menú del día

- ¿En dónde estamos?
- Repaso DSE
- ***Consideraciones y enunciado***

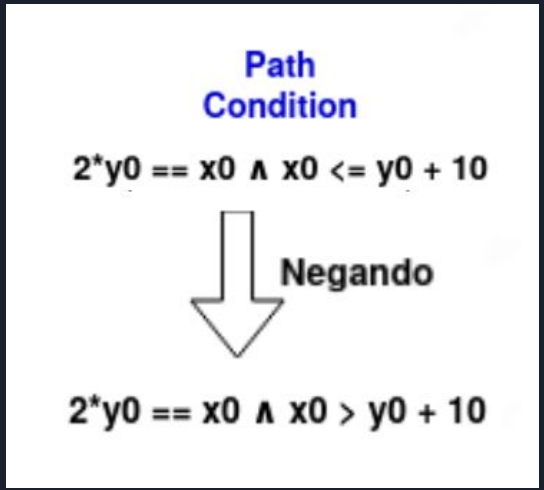
Consideraciones | *Soundness y Completeness*

- **Soundness:** Si DSE termina y no alcanzó un error => el programa no puede alcanzar el error en ninguna ejecución.

- ***Completeness:*** Si DSE reporta un error => existe una ejecución real del programa que puede alcanzar ese error.

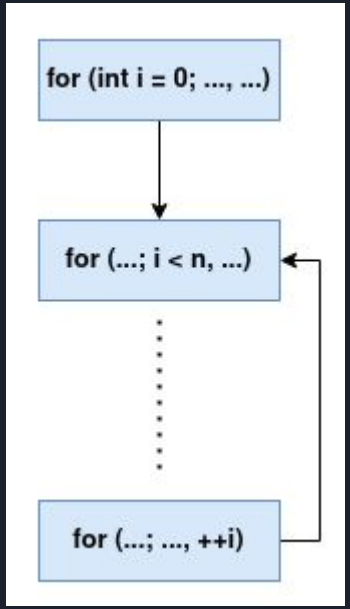
Consideraciones | *Algoritmo DSE*

- **<u>Navegación:</u>** Para elegir la próxima path condition a explorar usamos el algoritmo visto en la teórica.
    - ***<u>DFS:</u>*** Backtracking + negar la branch condition hasta encontrar alguna satisfacible.



Consideraciones | *Loops*

- **For:** ¿Cómo lo modelo para explorarlo?
    - Pensar como lo modelariamos en el grafo de control de flujo. (Define una branch condition)
    - Implícitamente define el árbol de cómputo subyacente.



Enunciado

Al enunciado!

Consideraciones | *¿A dónde apuntar del informe?*

- Breve introducción teórica de la técnica
- Breve explicación sobre los resultados obtenidos en los ejercicios de DSE
- Comparaciones entre DSE y Randoop

Consideraciones | *Resolución de ejercicios*

¿Qué formato uso?

- Para cada iteración
  - Input concreto utilizado

```
int f(int z) {
  if (z == 12) {
    return 1;
  } else {
    return 2;
  }
}
```

Iteración	Input Concreto	Condición de Ruta	Especificación para Z3	Resultado Z3
1	z=0	$z_0 \neq 12$	(assert (= z 12))	$z_0 = 12$
2	z=12	$z_0 = 12$	END	END

Consideraciones | *Resolución de ejercicios*

¿Qué formato uso?

- Para cada iteración
  - Input concreto utilizado
  - Condición de ruta resultante ***sin modificar***

```
int f(int z) {
  if (z == 12) {
    return 1;
  } else {
    return 2;
  }
}
```

Iteración	Input Concreto	Condición de Ruta	Especificación para Z3	Resultado Z3
1	z=0	$z_0 \neq 12$	(assert (= z 12))	$z_0 = 12$
2	z=12	$z_0 = 12$	END	END

Consideraciones | *Resolución de ejercicios*

¿Qué formato uso?

- Para cada iteración
  - Input concreto utilizado
  - Condición de ruta resultante ***sin modificar***
  - Especificación para Z3 ***del nuevo camino a explorar***

```
int f(int z) {
  if (z == 12) {
    return 1;
  } else {
    return 2;
  }
}
```

Iteración	Input Concreto	Condición de Ruta	Especificación para Z3	Resultado Z3
1	z=0	$z_0 \neq 12$	(assert (= z 12))	$z_0 = 12$
2	z=12	$z_0 = 12$	END	END

Consideraciones | *Resolución de ejercicios*

¿Qué formato uso?

- Para cada iteración
  - Input concreto utilizado
  - Condición de ruta resultante ***sin modificar***
  - Especificación para Z3 ***del nuevo camino a explorar***
  - Los valores resueltos por Z3

```
int f(int z) {
  if (z == 12) {
    return 1;
  } else {
    return 2;
  }
}
```

Iteración	Input Concreto	Condición de Ruta	Especificación para Z3	Resultado Z3
1	z=0	$z_0 \neq 12$	(assert (= z 12))	$z_0 = 12$
2	z=12	$z_0 = 12$	END	END

Bibliografía

- The Fuzzing Book (https://www.fuzzingbook.org/) by Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler
  - Capítulo IV - Semantical Fuzzing: Symbolic Fuzzing
  - Capítulo IV - Semantical Fuzzing: Concolic Fuzzing
    - Para lo que no vimos -> ***sección: SMT-Solvers***

- Artículos adicionales
  - SAGE: Scalable Automated Guided Execution (https://patricegodefroid.github.io/public_psfiles/cacm2012.pdf)
  - CUTE: A Concolic Unit Testing Engine for C (http://mir.cs.illinois.edu/marinov/publications/SenETAL05CUTE.pdf)

¿Preguntas?