IKOS Analyzer

Antonio Vélez Estévez, Universidad de Cádiz.

¿Qué es IKOS?

IKOS es una herramienta creada por la NASA para realizar el análisis formal de programas, este análisis formal se basa en la *teoría de la interpretación abstracta*.

Teoría de la interpretación abstracta

Es una teoría basada en la propiedad matemática de **solidez**.

Solidez

En lógica matemática, un sistema lógico tiene la propiedad de solidez si y solo si cada fórmula que se pruebe en el sistema es lógicamente válida respecto de la semántica del sistema.

Aplicación de la interpretación abstracta

Un software se puede entender como un sistema puramente lógico. Las aplicaciones concretas de esta teoría son el análisis estático formal y la extracción de información automática sobre las posibles ejecuciones del programa.

Para que podamos compilar IKOS necesitamos las siguientes dependencias:

- Un compilador de C++ que soporte C++14 **(gcc >= 4.9.2 o clang >= 3.4)**.
- **CMake >= 3.4.3**.
- **GMP >= 4.3.1**.
- **Boost >= 1.55**.
- **Python 2 >= 2.7.3 or Python 3 >= 3.3**.
- **SQLite >= 3.6.20**.
- **LLVM y Clang 9.0.x**.

Se va a proporcionar una máquina virtual con estas dependencias instaladas e IKOS listo para la práctica.

Para usar IKOS tenemos dos opciones:

- Usar el comando **`ikos [options] file[.c|.cpp|.bc|.ll]`**.
- Usar el comando **`ikos-scan make`** sobre un directorio con un makefile de un proyecto.

La diferencia entre el primero y el segundo es que el primero solo analiza un solo fichero y el segundo un proyecto completo.

```c
// Para la io (input/output).
#include <stdio.h>

char toUpper(char c) {
 if(c >= 'a' && c <= 'z')
    return c - 32;
 return c;
}

int main(int argc, char**argv)
{
   char str[] = "Hola que tal";

   for(size_t i = 0; i  < sizeof(str); ++i)
        str[i] = toUpper(str[i]);

   printf("%s", str);
}
```

```
[*] Compiling prueba1.c
[*] Running ikos preprocessor
[*] Running ikos analyzer
[*] Translating LLVM bitcode to AR
[*] Running liveness analysis
[*] Running widening hint analysis
[*] Running interprocedural value analysis
[*] Analyzing entry point 'main'
[*] Checking properties for entry point 'main'

# Time stats:
clang        : 0.344 sec
ikos-analyzer: 0.055 sec
ikos-pp      : 0.102 sec

# Summary:
Total number of checks                : 25
Total number of unreachable checks    : 0
Total number of safe checks           : 25
Total number of definite unsafe checks: 0
Total number of warnings              : 0

The program is SAFE

# Results
No entries.
```

```c
#ifndef MATRIZ_H
#define MATRIZ_H

struct Matriz {
  double** datos;
  int filas;
  int columnas;
};
typedef struct Matriz Matriz;

Matriz* matriz_inicializar(int filas, int columnas);
double matriz_get_elto(Matriz* matriz, int fila, int columna);
void matriz_set_elto(Matriz* matriz, int fila, int columna, double valor);
void matriz_liberar(Matriz* matriz);

#endif // MATRIZ_H
```

```
Matriz* matriz_inicializar(int filas, int columnas)
{
  Matriz* matriz = (Matriz*) malloc(sizeof(Matriz));

  matriz->filas = filas;
  matriz->columnas = columnas;
  matriz->datos = (double**) malloc(filas * sizeof(double*));

  for(int fila = 0; fila < filas; ++fila)
    matriz->datos[fila] = (double*) malloc(columnas * sizeof(double));

  return matriz;
}
```

```
// Programa para multiplicar dos matrices.
#include "matriz.h"
#include <stdio.h>

Matriz* multiplicarMatrices(Matriz* matriz1, Matriz* matriz2)
{
  Matriz* resultado = matriz_inicializar(matriz1->filas, matriz2->columnas);
  for(int i = 0; i < resultado->filas; ++i)
    for(int j = 0; j < resultado->columnas; ++j)
      {
        double suma = 0;
        for(int k = 0; j < matriz1->columnas; ++k)
          suma += matriz1->datos[i][k] * matriz2->datos[k][j];
        resultado->datos[i][j] = suma;
      }
  return resultado;
}
...
```

```
[*] Running ikos prueba2.bc -o prueba2.db
[*] Running ikos preprocessor
[*] Running ikos analyzer
[*] Translating LLVM bitcode to AR
[*] Running liveness analysis
[*] Running widening hint analysis
[*] Running interprocedural value analysis
[*] Analyzing entry point 'main'
[*] Checking properties for entry point 'main'

# Time stats:
ikos-analyzer: 0.030 sec
ikos-pp      : 0.009 sec

# Summary:
Total number of checks                : 141
Total number of unreachable checks    : 9
Total number of safe checks           : 98
Total number of definite unsafe checks: 0
Total number of warnings              : 34

The program is potentially UNSAFE

# Results
Report is too big (> 15 entries)
```

¿Por qué?

IKOS nos da este reporte diciendo que el programa es potencialmente inseguro porque:

- Hay un error de programación en un bucle. Se produce un bucle infinito y por eso nos da 9 comprobaciones de bloques de código inalcanzables.
- Hay 34 avisos. Entre ellos muchos relacionados con posibles punteros nulos, posibles dobles liberaciones de memoria... Todo esto hay que considerarlo para comprobar que el programa es seguro.

Limitaciones

Una de las limitaciones más destacables de IKOS Analyzer es que es incapaz de analizar código concurrente.

Muchas gracias.