Universidad del Valle de Guatemala
Facultad de Ingeniería
Ingeniería en Ciencia de la Computación y Tecnologías de la Información

# Laboratorio No. 11

## EDUARDO RAMIREZ 19946

Para los siguientes incisos, siga estas instrucciones:

- Entregue un archivo con extensión .pdf con todo su procedimiento y sus respuestas. Respuestas sin procedimiento tendrán una ponderación de 0 puntos.
- Esta actividad es individual.

**Actividad única (100%) –** En clase discutimos sobre decibilidad e indecibilidad y hablamos acerca de cómo podemos hacer que problemas en la industria que bien podrían ser indecidibles o que podrían ser problemas NP se "sientan" como problemas decidibles o con tiempo polinomial.

Una herramienta común utilizada en la industria con este fin son los SMTs o Satisfiability Modulo Theories solvers, que son herramientas de software que nos ayudan a determinar si determinadas fórmulas matemáticas son satisfactorias o no.

La idea es, en esencia, transformar problemas de la vida real a fórmulas o enunciados matemáticos para luego ser resueltos con SMT solvers.

Este laboratorio es más como una guía de investigación/aprendizaje para que usted lea un poco más acerca del uso en la industria de los SMT solvers y entienda el impacto con los temas vistos en el curso.

1. Lea el siguiente artículo de Amazon Web Services que explica cómo esta empresa utiliza razonamiento automatizado para ayudar a sus usuarios a obtener una mejor seguridad en la configuración de sus políticas de IAM: https://aws.amazon.com/blogs/security/protect-sensitive-data-in-the-cloud-with-automated-reasoning-zelkova/
    a. No necesita comprender mucho de AWS, lo importante es analizar el uso que se le da a los SMTs en este caso para resolver un problema que de lo contrario sería indecidible.
2. Luego, vea el siguiente video que explica más a fondo el uso de estas técnicas por parte de AWS y que explica la teoría detrás de estas técnicas:
    a. https://www.youtube.com/watch?v=U40bWY6oVtU
    b. Puede colocar el video en velocidad rápida, para avanzar más rápidamente si así lo desea.
3. Con base en la lectura y video anterior, así como con un poco más de investigación responda:
    a. ¿Qué es un SMT?

        Un SMT es una herramienta que ayuda a resolver si un conjunto de fórmulas lógicas puede ser verdadero al mismo tiempo. Imagina que tienes una serie de condiciones y quieres saber si es posible que todas se cumplan juntas. Los SMT solvers son programas que usan lógica matemática para verificar esto. Se utilizan en muchas áreas como la verificación de software, optimización y seguridad informática.

    b. ¿Cómo utiliza AWS los SMTs para potenciar la seguridad de sus clientes para analizar sus políticas de seguridad?

        AWS usa SMT solvers a través de una herramienta llamada Zelkova para revisar y analizar políticas de seguridad, como las de IAM y S3. Zelkova convierte estas políticas en fórmulas matemáticas precisas y las revisa con SMT solvers para ver si hay situaciones en las que la seguridad podría fallar.

Gracias a esto, AWS puede avisar a los usuarios si sus políticas permiten un acceso que no deberían, ayudándolos a proteger mejor sus datos y recursos.

c. Brinde 3 ejemplos adicionales del uso de SMT solvers en la industria y explíquelos brevemente.

**Verificación de software**: Las empresas de tecnología usan SMT solvers para revisar el código y asegurarse de que no tenga errores graves. Esto ayuda a prevenir problemas de seguridad y fallos en el funcionamiento del software.

**Optimización de rutas de entrega**: En logística, los SMT solvers se usan para planear rutas de entrega que sean más eficientes, respetando restricciones como tiempos de entrega y capacidad de vehículos.

**Diseño de circuitos electrónicos**: En la creación de hardware, los SMT solvers ayudan a verificar que los diseños de circuitos sean correctos antes de fabricarlos. Esto previene errores costosos en las etapas de producción.

4. Investigue acerca de programación funcional y responda:
   a. ¿Qué es la programación funcional?

La programación funcional es un estilo de programación donde se usan funciones como piezas clave del código. Se centra en escribir funciones que siempre producen el mismo resultado para las mismas entradas (llamadas funciones puras) y no cambian el estado de otras partes del programa. Esto hace que el código sea más fácil de entender y depurar.

   b. ¿Cómo se relaciona la programación funcional con el cálculo Lambda y con las máquinas de Turing?

El cálculo Lambda es una forma matemática de representar funciones y sirve como base para la programación funcional. Fue creado para entender cómo las funciones pueden ser usadas y evaluadas. Las máquinas de Turing son un modelo de cómo funcionan las computadoras de forma abstracta y pueden resolver cualquier cosa que se pueda programar. El cálculo Lambda y las máquinas de Turing tienen el mismo poder en términos de lo que pueden calcular, lo que significa que cualquier programa que puedas escribir con una máquina de Turing también se puede escribir usando el cálculo Lambda.

   c. Proporcione 5 ejemplos de lenguajes de programación funcional.

**OCaml**: Un lenguaje que combina características funcionales con programación orientada a objetos e imperativa.

**F#**: Un lenguaje funcional de la plataforma .NET que se integra bien con otros lenguajes como C#.

**Erlang**: Diseñado para sistemas distribuidos y de alta disponibilidad; ideal para aplicaciones en tiempo real.

**Scheme**: Un dialecto de Lisp que es simple y adecuado para aprender principios de programación funcional.

**Clojure**: Un lenguaje funcional moderno que se ejecuta en la JVM y es conocido por su enfoque en la inmutabilidad y la programación concurrente.

5. Investigue acerca de Z3, un SMT de functional programming desarrollado por Microsoft.
    a. Utilice el siguiente Playground online para aprender lo básico sobre Z3: https://jfmc.github.io/z3-play/

b. También puede consultar la documentación oficial de Microsoft sobre Z3: https://microsoft.github.io/z3guide/docs/logic/intro
c. Luego lea la documentación que explica acerca de satisfacibilidad y validez con Z3: https://microsoft.github.io/z3guide/docs/logic/propositional-logic#satisfiability-and-validity
d. Responda: ¿Cuál es el acercamiento que debe tomar con Z3 para probar si una fórmula determinada es siempre válida?

   Para comprobar si una fórmula es siempre válida usando Z3, primero se niega la fórmula y se revisa si esta versión negada es insatisfacible. Si Z3 dice que la fórmula negada no se puede satisfacer, entonces la fórmula original es válida en todos los casos posibles.

e. Analice el siguiente fragmento de código de Z3 y responda:

```
(define-fun mistery ((x Int)) Bool
  (and
    (> x 1)
    (not (exists ((z Int) (y Int))
      (and (< y x) (< z x) (> y 1) (> z 1) (= x (* y z)))))))
(assert (mistery 11))
(check-sat)
```

   i. ¿Cuál es el resultado luego de ejecutar este código en uno de los Playgrounds proporcionados? *Revise que ha copiado bien el código que intencionalmente ha sido colocado como imagen*.

      La función comprueba si un número x puede ser dividido en dos números más pequeños y y z, siguiendo ciertas reglas. Si no es posible, devuelve false. Cuando se prueba con x = 11, el resultado es que no hay números y y z que cumplan con todas las reglas, por lo que la función responde que la afirmación es insatisfactoria para ese caso

   ii. Pruebe la función con $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, \dots\} \in \mathbb{Z}^!$ y responda: ¿qué hace esta función?

Para 1: La función devolverá false porque x=1x = 1x=1 no cumple la condición x>1x > 1x>1.

Para 2: La función devolverá true porque 2 es un número primo.

Para 3: La función devolverá true porque 3 es un número primo.

Para 4: La función devolverá false porque 4=2×24 = 2 \times 24=2×2, es decir, tiene factores 222 y 222 mayores que 1 y menores que 4.

Para 5: La función devolverá true porque 5 es un número primo.

Para 6: La función devolverá false porque 6=2×36 = 2 \times 36=2×3, tiene factores 222 y 333 mayores que 1 y menores que 6.

Para 7: La función devolverá true porque 7 es un número primo.

Para 8: La función devolverá false porque 8=2×48 = 2 \times 48=2×4, tiene factores 222 y 444 mayores que 1 y menores que 8.

Para 9: La función devolverá false porque 9=3×39 = 3 \times 39=3×3, tiene factores 333 y 333 mayores que 1 y menores que 9.

Para 10: La función devolverá false porque 10=2×510 = 2 \times 510=2×5, tiene factores 222 y 555 mayores que 1 y menores que 10.

Para 11: La función devolverá true porque 11 es un número primo.

Para 12: La función devolverá false porque 12=2×612 = 2 \times 612=2×6, tiene factores 222 y 666 mayores que 1 y menores que 12.

Para 13: La función devolverá true porque 13 es un número primo.

La función mistery(x) determina si un número x es primo. Si x es un número primo, la función devolverá true, de lo contrario, devolverá false.