FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

PROGRAMA DE CURSO

Código	Nombre
CC7126	ANÁLISIS Y VERIFICACIÓN DE PROGRAMAS
Nombre en Inglés	
PROGRAM ANALYSIS AND VERIFICATION	

SCT	Unidades Docentes	Horas de Cátedra	Horas Docencia Auxiliar	Horas de Trabajo Personal
6	10	3	1.5	5.5

Requisitos	Carácter del Curso
Para el pregrado, el requisito será: CC4101 Lenguajes de Programación / Autorización El curso no presenta ningún requisito adicional, aunque será de gran ayuda cierto grado de madurez matemática por parte del alumno, en particular en el área de álgebra.	Electivo para Magister y Doctorado en Computación. Electivo avanzado para pregrado.

Resultados de Aprendizaje

Al término del curso, el estudiante tendrá un dominio general sobre las diferentes técnicas empleadas en la análisis y verificación formal de programas. El estudiante estará familiarizado con los fundamentos matemáticos subyacentes a cada una de dichas técnicas, reconociendo, en particular, sus alcances y limitaciones. El estudiante estará familiarizado, además, con los algoritmos y estrategias tradicionalmente utilizados para mecanizar dichas técnicas.

El estudiante que apruebe el curso manejará las nociones de:

- **Semántica formal de programas:** reconocerá dos estrategias clásicas para describir *rigurosamente* el significado o efecto de un programa.
- **Verificación formal de programas:** será capaz de *demostrar* (en el sentido matemático) la corrección funcional de un programa mediante el uso de lógica de Hoare y razonar también sobre programa que manipulan estructuras de datos dinámicas mediante el uso de lógica de separación.
- **Análisis estático de programas:** será capaz de razonar formalmente sobre propiedades más generales de los programas, de manera completamente automática, y en tiempo de compilación.

Metodología Docente	Evaluación General
Clases expositivas del profesor de cátedra que combinan el uso de diapositivas y la pizarra para la presentación del material. Para promover un proceso de aprendizaje interactivo, durante las clases de cátedra se presentarán regularmente mini-ejercicios para que los alumnos los piensen individual o grupalmente, y luego se discutirá colectivamente su solución. Las clases auxiliares, opcionales, serán clases prácticas de ayuda para la realización de las tareas.	Durante el curso se realizará una evaluación continua a través de tareas y una presentación oral: • **Tareas:** al final de cada unidad temática habrá una tarea individual, y sus notas se promediarán a partes iguales para obtener la *nota de tareas* (NT). • **Presentación:** el alumno deberá estudiar de manera independiente un tema relacionado al curso y dar una presentación oral al respecto, obteniendo una *nota de presentación* (NP). Los temas serán propuestos por el profesor de cátedra.

	Para aprobar el curso, las notas de tareas y presentación deben ser ambas mayores o iguales a 4.0. En particular, si la nota de tareas es igual o superior a 5.0, el alumno se eximirá de rendir el examen final y su nota final será **NF = 0.8 x NT + 0.2 x NP**. Por el contrario, si la nota de tareas es inferior a 5.0 (y mayor o igual a 4.0), el alumno deberá rendir un examen final, y su nota final será **NF = 0.4 x NT + 0.4 x NE + 0.2 x NP**, donde NE representa su nota de examen.

Unidades Temáticas

Número	Nombre de la Unidad	Duración en Semanas
1	Introducción	1

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Testing versus verificación formal de programas. Semántica formal de programas: variantes operacional y denotacional. Técnicas de análisis y verificación formal de programas: lógica de Hoare, análisis de flujo de datos, interpretación abstracta, sistemas de tipos.	Comprender el rol del análisis y verificación formal de programas en el contexto del desarrollo de software. Entender la semántica formal de programas como pilar en el desarrollo de técnicas de análisis y verificación formal. Reconocer el conjunto de técnicas clásicas de análisis y verificación formal.	[2] Cap 1 [5] Cap 1 [3] Cap 1-2

Número	Nombre de la Unidad	Duración en Semanas
2	Semántica operacional	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Relación de evaluación de expresiones. Relación de ejecución de comandos. Variantes de las relaciones: semántica *small-step* y *big-step*. Pruebas por inducción estructural y sobre derivaciones.	Interpretar un programa como una secuencia de cómputos (transiciones sobre una máquina abstracta). Comprender las limitaciones de las pruebas por inducción estructural y la necesidad de la inducción sobre derivaciones.	[1] Cap 2-4 [2] Cap 2-3

Número	Nombre de la Unidad	Duración en Semanas
3	Semántica denotacional	2

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Órdenes parciales completos (cpo's), continuidad, Teorema de Punto Fijo de Kleene. Función semántica sobre expresiones. Función semántica sobre comandos. Correspondencia entre semántica denotacional y operacional para un lenguaje imperativo simple.	Interpretar un programa como un elemento (denotación) dentro de un dominio apropiado. Entender los requerimientos matemáticos que hacen a un dominio apropiado para dicho fin. Entender la semántica denotacional de construcciones recursivas como un problema de punto fijo.	[1] Cap 5 [2] Cap 5

Número	Nombre de la Unidad	Duración en Semanas
4	Lógica de Hoare	2.5

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Corrección parcial y total de programas; juicios de la lógica de Hoare. Sistema de pruebas; corrección y completitud relativa. Mecanización del proceso de verificación: VCGen	Verificar la corrección funcional de un programa a través de un proceso deductivo (derivación sobre una lógica de programas). Comprender las limitaciones de dicho enfoque. Comprender cómo mecanizar el proceso de verificación.	[1] Cap 6-7 [2] Cap 9 [3] Cap 6

Número	Nombre de la Unidad	Duración en Semanas
5	Lógica de Separación	1.5

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Nociones de store y heap. Extensión del lenguaje de aserciones; nueva interpretación de los juicios. Extensión del sistema de pruebas.	Razonar sobre propiedades de corrección funcional *y* seguridad de memoria sobre programas que manipulan el heap (extendiendo adecuadamente la lógica de Hoare estudiada en la Unidad 4). Razonar localmente sobre programas a través de una regla de marco (*frame rule*).	[4] Cap 3 [7]

Número	Nombre de la Unidad	Duración en Semanas
6	Análisis de flujo de datos	3

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Análisis de programa clásicos basados en flujo de datos. Framework monótono. Algoritmos *Worklist* y *Meet over All Paths* para calcular el resultado de los análisis. Aproximaciones de puntos fijos a través de las técnicas de *widening* y *narrowing*.	Modelar diversos análisis de programas (intraprocedurales) como un problema de flujo de datos. Modelar el flujo de datos de un programa a través de un sistema de ecuaciones y resolver dicho sistema usando un algoritmo de iteración de punto fijo. Reconocer un framework general para especificar y calcular el resultado de análisis de programas basados en flujo de datos.	[5] Cap 2 y 4.2

Número	Nombre de la Unidad	Duración en Semanas
7	Interpretación abstracta	3

Contenidos	Resultados de Aprendizajes de la Unidad	Referencias a la Bibliografía
Pruebas ad-hoc de la corrección de análisis de programas; limitaciones. Conexiones de Galois; semántica concreta y abstracta de programas. Operaciones y análisis inducidos por conexiones de Galois (sobre dominios abstractos). Estudio de algunos dominios de abstracción clásicos.	Razonar sobre el comportamiento de un programa a través de una aproximación de su semántica real. Utilizar conexiones de Galois para desarrollar análisis de programas correctos por construcción de manera sistemática.	[5] Cap 4 [6]

Bibliografía

Referencia principal del curso:

[1] G. Winskel, *The Formal Semantics of Programming Languages*, MIT Press, 1993.

[2] F. Nielson et al*., Semantics with Applications: An Appetizer*, Springer, 2007.

[3] J. B. Almeida et al., *Rigorous Software Development: An Introduction to Program Verification*, Springer, 2011.

[4] A. Appel et al., *Program Logics for Certified Compilers*, Cambridge University Press, 2014.

[5] F. Nielson et al., *Principles of Program Analysis*, Springer, 2015.

FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Material complementario:

[6] P. Cousot, *A Tutorial on Abstract Interpretation*, VMCAI'05 Industrial Day on Automatic Tools for Program Verification, 2005.
Disponible online: https://homepage.cs.uiowa.edu/~tinelli/classes/seminar/

[7] P.W. O'Hearn, *A Primer on Separation Logic (and Automatic Program Verification and Analysis)*, Notas de la Escuela de Verano Marktoberdorf 2011, 2012.
Disponible online: http://www0.cs.ucl.ac.uk/staff/p.ohearn/papers/

Vigencia desde:	Otoño 2018
Elaborado por:	Federico Olmedo