

	TEORIA DE LA COMPUTACION		3.4.215
Departamento al que pertenece		Director	
Tecnología Informática		Ing. Anibal Freijo	
Carga horaria		Fecha de aprobación en el Consejo de Facultad y N° de Acta	
68 horas		17/08/2021 CF FAIN N° 444	
Carrera(s) en la que se dicta			Código(s) Carrera(s)
Ingeniería en Informática			1621
Código(s) Correlativa(s) Precedente(s)	Código(s) Correlativa(s) Subsiguiente(s)	Código(s) Carrera(s)	
3.4.077		1621	
Firmas			
Aprobación del Director de Departamento emisor.		Aprobación Decano(s)	
<div data-bbox="443 1350 587 1547" data-label="Text">  </div> <div data-bbox="311 1556 724 1659" data-label="Text"> <p>Ing. Anibal Freijo Director del Departamento de Informática Facultad de Ingeniería y Ciencias Exactas Fundación UADE</p> </div>		<div data-bbox="1002 1435 1177 1570" data-label="Text">  </div> <div data-bbox="885 1576 1292 1688" data-label="Text"> <p>Dr. Federico Prada Decano a cargo Facultad de Ingeniería y Cs. Exactas Fundación UADE</p> </div>	

I. Fundamentos de la materia

La teoría de la computación es un área de la matemática y la informática que esencialmente trata de establecer qué problemas pueden ser resueltos y cuán eficientemente. Esta disciplina puede dividirse entre computabilidad y complejidad. En esta materia el énfasis está puesto en la computabilidad. Para ello es necesario comprender los modelos matemáticos subyacentes al proceso de computación: autómatas, gramáticas formales. Finalmente se darán las nociones de corrección de programas (total y parcial) y complejidad.

Esta materia toma como base directa los conocimientos impartidos en la materia Matemática Discreta y como base indirecta la experiencia de los alumnos en las materias de programación. Se retoman algunos ejemplos de estas últimas vistos desde el punto de vista de la eficiencia computacional (por ejemplo, la complejidad de los algoritmos de ordenamiento.) Se recalca la importancia de algunos métodos en áreas aplicadas (por ejemplo, el uso de autómatas para la construcción de compiladores y el uso de redes de Petri para la especificación de determinados requerimientos.)

II. Objetivos

Que el estudiante sea capaz de:

- Conocer los fundamentos teóricos que subyacen los procesos de computación. En particular, deben tener una cabal comprensión de los conceptos matemáticos fundamentales de la computación (autómatas, máquinas de Turing, lenguajes formales) y de su relación con los procesos de computación reales.
- Conocer las nociones de corrección parcial y total de programas.
- Comprender el concepto de decidibilidad y computabilidad, así como realizar análisis de complejidad de algunos procesos.
- Efectuar cálculos simples de complejidad.
- Estar en condiciones de diseñar y construir un parser simple (por ejemplo, para un fragmento del lenguaje Java)
- Estar en condiciones de utilizar modelos como autómatas y redes de Petri para la especificación de algunos problemas del mundo real.

III. Contenidos

Contenidos mínimos

Lenguajes. Expresiones y conjuntos regulares. Autómatas finitos determinísticos. Autómatas no determinísticos. Gramáticas. Jerarquía de Chomsky. Gramáticas regulares y autómatas finitos. Gramáticas libres de contexto y lenguajes. Análisis determinísticos. Máquinas de Turing. Decidibilidad. Computabilidad. Complejidad. Sintaxis y semántica formales de lenguajes de programación. Redes de Petri y su uso para modelado.

Contenidos conceptuales

Unidad I: Introducción.

Elementos fundamentales: lógica, grafos, autómatas, complejidad, computabilidad. Inducción matemática y estructural

Unidad II: Autómatas y lenguajes. Gramáticas formales

Autómatas de estados finitos. Definición de computación. Expresiones regulares. Determinismo y no determinismo. Equivalencia entre NFA y DFA. Reducción de autómatas. El lema de inflado (pumping lemma.) Gramáticas regulares y autómatas finitos. Lenguajes y expresiones regulares. Lenguajes no regulares. Gramáticas libres de contexto y dependientes de contexto. Jerarquía de Chomsky. Autómatas con pila (pushdown automática.)

Unidad III: Semántica y corrección de programas

Semántica de lenguajes de computación procedurales y funcionales. Corrección parcial y total. Condiciones previas más débiles y condiciones posteriores más fuertes (weakest preconditions y strongest postconditions.) Especificaciones formales. Lógica de Hoare.

Unidad IV: Computabilidad

Máquinas de Turing. Definición de algoritmo y computabilidad. El problema de la detención de la máquina de Turing. La tesis de Church-Turing. Decidibilidad. Máquinas de Turing y lenguajes sensitivos al contexto.

Unidad V: Redes de Petri

Redes de Petri. Resultados fundamentales y su utilización en modelado. Análisis. El problema de la alcanzabilidad

Unidad VI: Complejidad

Nociones fundamentales de complejidad. Medida de la complejidad. La notación O. Las clases P y NP. Problemas NP-completos. Complejidad espacial.

Contenidos procedimentales

- Realización de pruebas simples por inducción.
- Construcción y análisis de autómatas finitos. Determinación de las categorías de lenguajes.
- Realización de pruebas de corrección simples.
- Determinación de algunos problemas de decisión simples.
- Modelado de problemas simples (conurrencia, exclusión mutua) con redes de Petri.
- Determinación de complejidad en algunos casos simples.

Contenidos actitudinales

- Comprensión de la importancia de las pruebas. Comprensión de la extensión de inducción sobre los números naturales a la inducción estructural.
- Comprensión de la diferencia entre lenguajes regulares y lenguajes libres de contexto. Comprensión de la estrategia de solución de la equivalencia entre autómatas determinísticos y no determinísticos.
- Comprensión del objetivo de la definición de una semántica formal. Diferencia entre corrección parcial y total.
- Comprensión del problema de decidibilidad.
- Comprensión de la utilidad de las redes de Petri como herramienta de modelado.
- Concientización de la importancia de la determinación de la complejidad de un problema de programación.

IV. Estrategias de enseñanza

Exposiciones dialogadas para la enseñanza de los conceptos, combinando exposición docente con participación activa de los alumnos.

V. Recursos

- Aula tradicional con medios para la proyección de contenidos
- Software necesario para la realización de los trabajos prácticos

VI. Modalidad de Evaluación

La asignatura TEORIA DE LA COMPUTACION se aprueba con Promoción Directa.

Para aprobar la asignatura el alumno deberá cumplir con la aprobación de las instancias de evaluaciones parciales, que serán individuales y de tipo escrito y/u oral, y contar con un mínimo del 75% de asistencia a clase. En el caso de materias con posibilidad de promoción, los alumnos que obtuviesen una calificación mínima de 7 (siete) puntos en cada una de las evaluaciones parciales (sin rendir recuperatorio) obtendrán

la aprobación de la asignatura, cuya nota final consistirá en el promedio simple de las calificaciones de las evaluaciones parciales. Los alumnos que, habiendo aprobado las evaluaciones parciales con un mínimo de 4 (cuatro) puntos o bien reprobado una de las dos instancias evaluatorias parciales pero aprueben luego el recuperatorio correspondiente, no alcanzarán la promoción pero podrán rendir examen final de la asignatura en los 11 (once) turnos de exámenes finales consecutivos posteriores a la aprobación de la cursada. En este caso se consignará como nota final al promedio simple entre la nota de aprobación de la cursada (promedio de las calificaciones de las evaluaciones parciales aprobadas) y la calificación obtenida en el examen final regular. En el caso que el alumno haya aprobado las instancias de evaluación y no requiera recuperar, podrá optar por rendir el examen final regular en la fecha prevista para el examen recuperatorio o bien en la fecha prevista para el examen final regular (una de las dos). Los alumnos que rindan el examen final en la etapa de previos, la nota final a consignarse será exclusivamente la obtenida en dicha instancia de evaluación. Los actos de deshonestidad académica o cualquier situación de indisciplina serán sancionados según el régimen disciplinario correspondiente.

NORMAS DE SEGURIDAD: El trabajo en laboratorios y talleres debe llevarse a cabo respetando las normas de seguridad obligatorias. La aprobación de la cursada/materia estará sujeta al cumplimiento de las mismas, ya que son el principal factor de riesgo en las actividades de los alumnos, docentes, investigadores o técnicos.

Utilizar siempre los Equipos de Protección Individual que se requiera (consultar procedimientos o protocolos de trabajo), por ejemplo protección ocular (anteojos gafas/pantallas faciales), guantes de vinilo y guardapolvo.

VII. Bibliografía

Básica

HOPCROFT, John. MOTWANI, Rajeev y ULLMAN, Jeffrey. Introduction to Automata Theory, Languages and Computation. 2007. Boston : Pearson : Addison Wesley. xvii, 535 p. : il. ISBN 9780321455369

MANNA, Zohar. Mathematical Theory of Computation. 2003. New York : Dover. 448 p. ISBN 9780486432386

SIPSER, Michael. Introduction to the Theory of Computation. 2006. Boston: Course Technology. xix, 431 p. : il. ISBN 9780534950972

Complementaria

DAVIS, Martin. Computability and Unsolvability. New York : Dover, 1985. xxv, 248 p. ISBN 9780486614717

FLOYD, Robert. Assigning Meanings to Programs. 1967. Proceedings of Symposia in Applied Mathematics. Vol. 19, 19-31 p.

HOARE, Charles. An Axiomatic Basis for Computer Programming. 1969. Communications of the ACM. Vol. 12, 10, pp 576-580, 583.

LEWIS, Harry; PAPADIMITRIOU, Christos. Elements of the Theory of Computation. 1997. Prentice Hall. ISBN-10: 0132624788. ISBN-13: 978-0132624787

MURATA, Tadao. Petri Nets: Properties, Analysis and Applications. 1989. Proceedings of the IEEE. Vol. 77, 4, 541-580 pp.

PETERSON, James. Petri Nets. 1977. ACM Computing Surveys. Vol. 9, 3, 223-252 pp.

TENNENT, Charles. The Denotational Semantics of Programming Languages. 1976. Communications of the ACM. Vol. 19, 8, pp 438-453.

VIII. Cronograma

Clase	Actividad/contenido
1	Contenidos Generales de la Materia. Nociones fundamentales: alfabetos, cadenas, lenguajes. Autómatas de estados finitos (AF) y lenguajes reconocidos por un autómata. Diseño de autómatas y lenguajes regulares (nivel 3 en la jerarquía de Chomsky.)
2	Expresiones regulares. Algoritmo para construir un AFNDe a partir de una expresión regular. Algoritmo para obtener una expresión Regular a partir de un DFA. Práctica de autómatas Finitos y Expresiones Regulares.
3	Algoritmos autómatas finitos- Obtener una Autómata Finito no determinístico a partir de un AFNDe. Equivalencia de AFND y AFD. Reducción de autómatas. Obtención del autómata DFA con estados mínimos. Resolución de Ejercicios
4	Autómatas de Pilas (AP). Autómatas de Pilas Determinísticos y no Determinísticos. Resolución de Ejercicios de Autómatas de Pilas.
5	Gramáticas- Jerarquía de Chomsky. Gramáticas Regulares. Resolución Ejercicios Gramáticas Regulares
6	Gramáticas Libres de Contexto. Resolución de Ejercicios Gramáticas Libres de Contexto.
7	Máquinas de Turing determinísticas y no determinísticas. Autómatas Linealmente Acotados. Resolución de Ejercicios Máquinas de Turing.
8	Máquinas de Turing para Cálculo de Funciones. Resolución de Ejercicios de Máquinas de Turing.
9	Gramáticas Sensibles al Contexto. Resolución de Ejercicios Gramáticas sensibles al contexto
10	Clase de Consulta y Repaso General Previo
11	Parcial
12	Lemma Pumping. Verificación de Lenguajes Regulares. Resolución de Ejercicios Lemma Pumping
13	Introducción a los parsers. Gramáticas LL1 Resolución ejercicios LL1.
14	Redes de Petri. Definición y modelado de procesos. Resolución de Ejercicios
15	Complejidad. Las clases P y NP. Problemas NP-Completo.
16	Clase de Consulta y Repaso General
17	Recuperatorio y Final Regular
	Final Regular