

PROGRAMA DE ESTUDIOS: TEORÍA DE LA COMPUTACIÓN

PROTOCOLO

Fechas	Mes/año
Elaboración	05-2010
Aprobación	
Aplicación	09-2010

Clave			Semestre	5°		
Nivel	Licenciatura	X	Maestría		Doctorado	
Ciclo	Integración		Básico		Superior	
Colegio	H. y C.S.		C. y T.	X	C. y H.	

Plan de estudios del que forma parte:
Licenciatura en Ingeniería de Software

Propósito(s) general(es):
El estudiante aplicará las teorías de lenguajes formales, gramáticas y las máquinas de estados finitos, en el diseño e implementación de aplicaciones reales de software. Además podrá reconocer aquellos problemas que no puedan ser resueltos utilizando una computadora.

Carácter		Modalidad				Horas de estudio semestral (16 semanas)					
Indispensable	X	Seminario	X	Taller	Curso-taller	Con	Teóricas	72	Autónomas	Teóricas	56
						Docente	Prácticas			Prácticas	
Optativa *		Laboratorio		Clínica		Carga horaria semanal 4.5 + 3.5 = 8			Carga horaria semestral: 128		

Asignaturas Previas:	Asignaturas Posteriores:
Matemáticas Discretas, Estructuras de Datos y Programación Orientada a Objetos	Lenguajes de programación

Requerimientos para cursar la asignatura
Conocimientos y habilidades:
El estudiante debe tener conocimientos de matemáticas discretas, principalmente conjuntos y relaciones, e inducción y recursión; además debe conocer las estructuras de datos, su representación y programación, además de manejar un lenguaje de programación de alto nivel. Habilidades: Abstracción, análisis y creatividad.

Perfil deseable del profesor:
Maestría en Ciencias de la Computación, Ingeniería de Software o disciplinas afines. Además, experiencia docente y de investigación.

Academia responsable del programa:	Diseñador (es):
Informática	M en C. Araceli Liliana Reyes Cabello

INTRODUCCIÓN

La teoría de la computación se encarga de estudiar las limitaciones y capacidades fundamentales de las computadoras valiéndose de modelos matemáticos, que representan máquinas computacionales de estados finitos, con las cuales se formaliza el concepto de hacer cómputo. En esta asignatura se estudiarán los principales modelos de máquinas computacionales (autómatas con un número de estados finitos, autómatas de pila, autómatas linealmente acotados y máquinas de Turing); al mismo tiempo se revisarán los tipos de lenguajes que éstas máquinas reconocen, junto con los mecanismos que los generan (gramáticas), resaltando la correspondencia y la relación de jerarquía que existe entre éstos. Finalmente se discutirán nociones de decidibilidad e indecidibilidad relacionándolo con el análisis de complejidad.

El curso está estructurado en siete unidades:

La primera es una introducción al tema de los autómatas y lenguajes formales, junto con su historia. Se definen los conceptos básicos de los elementos con los que se va a trabajar como: símbolos, alfabetos, cadenas, lenguajes, gramáticas; y se presenta el proceso de generación de lenguajes. Posteriormente, en la segunda unidad se estudian los autómatas finitos, analizando la equivalencia entre el modelo determinista y el no determinista. Asimismo, se incursiona en el tema de los lenguajes regulares y sus propiedades, así como en las gramáticas regulares y expresiones regulares, que son el mecanismo que los genera. Al final de la unidad se revisa la relación que existe entre autómatas, lenguajes regulares y expresiones regulares. La tercera unidad está enfocada a revisar el modelo de autómata de pila, como una extensión de un autómata finito al cual se le agrega memoria en forma de pila, analizando los beneficios que existen al hacer esta extensión. Del mismo modo, se revisan los diferentes modelos de estos autómatas y si hay o no equivalencia que existe entre ellos. En la siguiente unidad se presentan los lenguajes y gramáticas libres de contexto, junto con las propiedades de éstos. Se revisan las diferentes transformaciones que se pueden realizar a una gramática de este tipo, y se demuestra la equivalencia entre los autómatas de pila y las gramáticas libres de contexto. La unidad cinco está dedicada al estudio de las Máquinas de Turing, incluyendo los autómatas linealmente acotados. Se presentan los diferentes tipos de Máquinas de Turing, resaltando a la máquina de universal de Turing como modelo universal de cómputo. El siguiente tema es la Jerarquía de Chomsky, en la cual se estudia la contención propia que existe entre los diferentes tipos de lenguajes; enfatizando que existe una amplia gama de lenguajes que escapan a esta clasificación. Además se prueba la equivalencia entre lenguajes generales y máquinas de Turing; y entre autómatas linealmente acotados y lenguajes dependientes del contexto. El último tema tiene como fin revisar el concepto de decidibilidad, para ello se muestran algunos ejemplos clásicos de problemas no decidibles, como el problema del paro. Concluyendo el tema con el teorema de Rice.

PROPÓSITOS GENERALES

El estudiante aplicará las teorías de lenguajes formales, gramáticas y las máquinas de estados finitos, en el diseño e implementación de aplicaciones reales de software. Además podrá reconocer aquellos problemas que no puedan ser resueltos utilizando una computadora

CONTENIDOS

TEMAS Y SUBTEMAS	PROPÓSITOS ESPECIFICOS
<p>1. Introducción</p> <p>1.1 Motivación 1.2 Conceptos básicos 1.3 Gramáticas y lenguajes formales 1.4 Árboles de derivación</p>	<p>Es estudiante definirá los conceptos centrales de la teoría de lenguajes formales y autómatas; y describirá el proceso para generar cadenas de un lenguaje formal.</p>
<p>2. Autómatas finitos y lenguajes regulares</p> <p>2.1 Descripción informal de los autómatas finitos 2.2 Autómatas finitos deterministas 2.3 Autómatas finitos no deterministas 2.4 Equivalencia 2.5 Minimización 2.6 Autómatas finitos con transiciones ϵ 2.7 Gramáticas tipo 3 y expresiones regulares 2.8 Relación entre autómatas finitos, gramáticas regulares y expresiones regulares 2.9 Propiedades de los lenguajes regulares 2.10 Aplicaciones</p>	<p>El estudiante construirá gramáticas regulares, expresiones regulares y autómatas finitos que generen y reconozcan lenguajes regulares, respectivamente. Además asociará a los autómatas finitos como analizadores léxicos.</p>
<p>3. Autómatas de pila</p> <p>3.1 Definición de los Autómatas de Pila 3.2 Aceptación por estado final 3.3 Aceptación por pila vacía 3.4 Transformación entre los dos tipos de autómatas. 3.5 Autómatas de pila con un solo estado 3.6 Aplicaciones: Traductores</p>	<p>El estudiante construirá autómatas de pila y para reconocer lenguajes libres de contexto, además de que los utilizará para traducir cadenas y para realizar análisis sintáctico.</p>
<p>4. Lenguajes Libres de Contexto</p> <p>4.1 Gramáticas Libres de Contexto 4.2 Simplificación de gramáticas libres de contexto. 4.3 Formas normales 4.4 Aplicaciones de la gramáticas libres de contexto 4.5. Equivalencia entre autómatas de pila y lenguajes libres de contexto 4.6 Propiedades de los lenguajes libres de contexto 4.7 Caracterización de los lenguajes que no son libres de contexto</p>	<p>El estudiante diseñará gramáticas libres de contexto; además demostrará la equivalencia que hay entre este tipo de gramáticas y los autómatas de pila.</p>

<p>5. Máquinas de Turing</p> <p>5.1 Motivación 5.2 Definición de la Máquina de Turing básica 5.3. Técnicas de programación para las Máquinas de Turing 5.4. Variantes de la Máquina de Turing 5.5 La Máquina de Turing Universal 5.6 Autómatas linealmente acotados</p>	<p>El estudiante construirá máquinas de Turing para reconocer lenguajes y descubrirá el poder computacional de este modelo matemático.</p>
<p>6. La jerarquía de Chomsky</p> <p>6.1 Lenguajes recursivos y recursivamente numerables 6.2 Gramáticas sin restricciones 6.3 Lenguajes generales y máquinas de Turing 6.4 Lenguajes dependientes del contexto y autómatas linealmente acotadas</p>	<p>El estudiante distinguirá las características de cada uno de los lenguajes formales y los alcances y limitaciones de cada uno de los autómatas asociados a éstos.</p>
<p>7. Introducción a la decidibilidad</p> <p>7.1. Definición de decidibilidad 7.2 El problema de detención (halting problema) 7.3. Reducción 7.4 Indecidibilidad en Máquinas de Turing 7.5 Otros problemas de indecidibilidad</p>	<p>El estudiante reconocerá que existen problemas que no pueden ser resueltos por una computadora.</p>

METODOLOGÍA PARA EL CURSO

La materia se impartirá por medio de exposiciones teóricas por parte del profesor, apoyándose en software libre que le permita ilustrar algunos de los temas. Asimismo se sugiere resolver ejercicios en clase junto con los estudiantes para reforzar el conocimiento, así como dinámicas grupales que tengan el mismo fin. Al final de cada tema exponer las aplicaciones reales en las que se ocupan los conceptos estudiados.

También se sugiere complementar ciertos aspectos del curso mediante el desarrollo de proyectos de programación realizados en algún lenguaje de alto nivel.

En las horas autónomas de estudio, el estudiante solucionará problemas teóricos que se plantearán en clase, y si es el caso, realizará prácticas o proyectos de programación.

EVALUACIÓN DIAGNÓSTICA

Se aplicará un examen escrito que permita evaluar los conocimientos que el estudiante tiene de conjuntos y relaciones, inducción y recursión, así como la resolución de problemas.

EVALUACIÓN FORMATIVA

Con el propósito de dar seguimiento al proceso de enseñanza aprendizaje, se propone aplicar tres evaluaciones escritas: la primera que incluya los temas expuestos en las dos primeras unidades, la segunda para evaluar la unidad tres y cuatro, la tercera que evalúe la unidad 5 y la 6. Para evaluar la última unidad se sugiere un trabajo escrito. Antes de cada evaluación escrita se sugiere dejar tareas a los estudiantes, preferentemente al iniciar cada tema, con el fin de que la resuelvan paulatinamente conforme se estudia el tema en clase. Posteriormente, se recomienda realizar un análisis con los estudiantes para evaluar los temas en los que se presenten problemas para realizar una revisión de los mismos. En el caso de que sean pocos los estudiantes que tengan dudas, éstas se tratarán en asesoría. Se pondrá especial atención en el cumplimiento de las tareas y la participación de los estudiantes en clase, ya que estas actividades ayudan a la formación del estudiante.

EVALUACIÓN DE CERTIFICACIÓN

Se proponen dos modalidades de certificación:

- 1) Portafolio: Que incluya tanto las tareas, evaluaciones y, si es el caso, proyectos que el estudiante deberá realizar a lo largo del semestre, y que son definidas por el profesor asignado a cada grupo.
- 2) El examen escrito final, elaborado por el comité de certificación.

En ambas modalidades se evaluarán tanto los conocimientos del estudiante de todos y cada uno de los temas que forman parte del programa de estudio, así como su habilidad para resolver problemas utilizando las herramientas expuestas durante el curso.

BIBLIOGRAFÍA

La bibliografía que se menciona a continuación es la básica para el estudiante y el profesor:

Núm.	Bibliografía	Temas en los que se recomienda
1.	Aho A.V., Lam M., Sethi S.R., and Ullman J.D. <i>Compilers: Compilers: Principles, Techniques & Tools</i> , Pearson Addison-Wesley, 2007	1,2,3,4,5,6,7
2.	Brookshear, J. Glenn. <i>Teoría de la Computación, Lenguajes Formales, Autómatas y Complejidad</i> . Addison Wesley Iberoamericana. 1993.	1,2,3,4,5,6,7
3.	Kozen, D.C., <i>Automata and Computability</i> , Springer-Verlag TELOS, 1997	1,2,3,4,5,6,7
4.	Linz, P. <i>An introduction to formal languages and automata</i> , D.C. Heath and Company, fourth edition, 2006.	1,2,3,4,5,6,7
5.	Hopcroft, J. E; Montwani, R., and Ullman, J. <i>Introduction to Automata Theory Language, and Computation</i> . Addison Wesley, third edition 2007.	1,2,3,4,5,6,7
6.	Sipser, M. <i>Introduction to theory of computation</i> , PWS Publishing Company, second edition, 2005.	1,2,3,4,5,6,7

OTROS RECURSOS

Proyector de video (Cañón), computadora que tenga instalado un compilador, editor de texto, depurador; preferentemente que tenga instalado un Entorno de Desarrollo Integrado (IDE).