PROGRAMA DE CURSO
TEORÍA DE LA COMPUTACIÓN

A. Antecedentes generales del curso:

Departamento	Ciencias de la computación					
Nombre del curso	Teoría de la computación		Código	CC3102	Créditos	6
Nombre del curso en inglés	*Theory of Computation*					
Horas semanales	Docencia	3	Auxiliares	1,5	Trabajo personal	5,5
Carácter del curso	Obligatorio	X		Electivo		
Requisitos	CC3101: Matemáticas discretas para la computación					

B. Propósito del curso:

El curso tiene como propósito que las y los estudiante adquieran los conceptos fundamentales de lenguajes formales, computabilidad y complejidad computacional, estableciendo así algunos límites de la computación.

Asimismo, el/la estudiante será capaz de clasificar lenguajes formales en la jerarquía de regulares, libres del contexto, decidibles, aceptables y no aceptables, así como la jerarquía de lenguajes P, NP, y NP-completos. Adquirirá herramientas formales para clasificar lenguajes, así como herramientas para resolver problemas prácticos que involucran describir y procesar lenguajes regulares y libres del contexto, particularmente, usando gramáticas y autómatas.

El curso tributa a las siguientes competencias específicas (CE) y genéricas (CG):

CE1: Analizar problemas computacionales, construir modelos, expresándolos en representaciones y lenguajes formales adecuados.

CE2: Analizar, diseñar y/o adaptar algoritmos y estructuras de datos que cumplan con las garantías requeridas de correctitud y eficiencia.

CG1: Comunicación académica y profesional

Comunicar en español de forma estratégica, clara y eficaz, tanto en modalidad oral como escrita, puntos de vista, propuestas de proyectos y resultados de investigación fundamentados, en situaciones de comunicación compleja, en ambientes sociales, académicos y profesionales.

C. Resultados de aprendizaje:

Competencias específicas	Resultados de aprendizaje
CE1	RA1: Distingue y expresa lenguajes regulares en base al uso de autómatas finitos y expresiones regulares, así como los lenguajes libres de contexto, usando autómatas de pila y gramáticas libres del contexto.
	RA2: Demuestra formalmente que un lenguaje es o no es regular o libre del contexto en base al lema de bombeo, teorema de bombeo y propiedades de clausura.
CE2	RA3: Diseña y programa soluciones computacionales a problemas de procesamiento de textos mediante el uso de autómatas, utilizando conceptos de lenguajes regulares y libres del contexto.
CE1	RA4: Utiliza Máquinas de Turing como un modelo de computación para representar, calcular funciones, decidir y aceptar lenguajes.
	RA5: Demuestra formalmente, mediante reducciones, que ciertos problemas no se pueden resolver computacionalmente, clasificándolos en decidibles, aceptables y no aceptables, extendiendo la validez de estas conclusiones a otros modelos de computación a partir de la tesis de Church.
CE2	RA6: Analiza y explica el concepto de complejidad computacional, el significado de las clases de problemas P y NP, demostrando la NP-completitud de problemas mediante el uso de reducciones.
Competencias genéricas	**Resultados de aprendizaje**
CG1	RA7: Produce textos de diversa extensión donde informa los resultados de la aplicación de conceptos de lenguajes formales para la solución de determinados problemas, considerando claridad y concisión en el desarrollo de sus ideas, argumentos o propuesta.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA1, RA2, RA3, RA7	Lenguajes Regulares	3 semanas
Contenidos		**Indicador de logro**	
1.1. Alfabetos, cadenas y lenguajes. Representación finita de lenguajes. 1.2. Autómatas finitos determinísticos. 1.3. Autómatas finitos no determinísticos. 1.4. Equivalencia entre ambos tipos de autómatas. 1.5. Expresiones regulares. 1.6. Equivalencia entre expresiones regulares y autómatas. 1.7. Propiedades de clausura y algorítmicas. 1.8. Lema del bombeo.		La/el estudiante: 1. Distingue y expresa lenguajes regulares, mediante el uso de autómatas finitos y expresiones regulares. 2. Demuestra que un lenguaje no es regular, usando el lema de bombeo. 3. Demuestra que un lenguaje es o no es regular, usando propiedades de clausura. 4. Diseña y programa soluciones computacionales simples para problemas prácticos de procesamiento de textos mediante el uso de autómatas finitos. 5. Reporta, de forma clara, en el contexto de las tareas, los resultados de la programación de las soluciones.	
Bibliografía de la unidad		[1] Cap. 2. [2] Cap. 1 y 2 (2.1 – 2.4). [4] Cap. 1.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
2	RA1, RA2, RA3, RA7	Lenguajes Libres del Contexto	3 semanas
Contenidos		**Indicador de logro**	
2.1. Gramáticas libres del contexto. 2.2. Autómatas de pila. 2.3. Equivalencia entre gramáticas y autómatas. 2.4. Propiedades de clausura y algorítmicas. 2.5. Teorema de bombeo. 2.6. Determinismo y parsing.		La/el estudiante: 1. Distingue y expresa lenguajes libre del contexto, usando autómatas de pila y gramáticas libres del contexto. 2. Demuestra que un lenguaje no es libre del contexto en base al teorema de bombeo. 3. Demuestra que un lenguaje es o no libre del contexto, usando propiedades de clausura. 4. Usa herramientas de parsing para resolver problemas prácticos de procesamiento de datos. 5. Informa y justifica, de manera clara, en el contexto de las tareas, los resultados obtenidos de la programación de las soluciones.	
Bibliografía de la unidad		[1] Cap. 3. [2] Cap. 3 (3.7 sólo parte). [4] Cap. 2.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA4, RA5	Máquinas de Turing	2,5 semanas
Contenidos		**Indicador de logro**	
3.1. Definición de Máquinas de Turing (MTs). Configuraciones y modelo de computación. 3.2. Modularización y solución de problemas más complejos usando MTs. Uso de MTs para decidir lenguajes, aceptar lenguajes, y calcular funciones. 3.3. Extensiones de MTs: múltiples cintas y otras. Simulaciones. 3.4. MTs no determinísticas y su simulación.		La/el estudiante: 1. Relaciona el formalismo de la Máquina de Turing con un modelo de computación mediante el uso de ejemplos. 2. Utiliza Máquinas de Turing básicas y extendidas, determinísticas y no determinísticas, para calcular funciones y reconocer lenguajes. 3. Clasifica lenguajes en decidibles y aceptables, mediante el uso de Máquinas de Turing con las cuales es posible identificarlos.	
Bibliografía de la unidad		[1] Cap. 4 [2] Cap. 4 (4.1 – 4.3 y 4.5) [4] Cap. 3	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
4	RA4, RA5	Computabilidad	3,5 semanas

Contenidos	Indicador de logro
4.1. La Máquina Universal de Turing. Simulación. 4.2. La Tesis de Church y otros formalismos. Equivalencia con el modelo RAM. 4.3. El problema de la detención. Lenguajes decidibles y aceptables. 4.4. Otros problemas indecidibles acerca de MTs. Reducción de problemas. 4.5. Gramáticas dependientes del contexto. Equivalencia con MTs. Enumerabilidad. 4.6. Problemas indecidibles acerca de gramáticas. Otros problemas indecidibles.	La/el estudiante: 1. Explica el funcionamiento, significado e importancia de la Máquina Universal de Turing, a partir de ejemplos. 2. Analiza ejemplos emblemáticos de lenguajes decidibles y aceptables, como es el problema de la detención. 3. Demuestra, mediante reducciones, que ciertos lenguajes no se pueden decidir o aceptar. 4. Analiza y argumenta cómo la Tesis de Church da validez universal a resultados demostrados con Máquinas de Turing y que es extendible su validez a otros modelos de computación.
Bibliografía de la unidad	[1] Cap. 5 [2] Cap. 4.6 y 5 [4] Cap. 4 y 5

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
5	RA3, RA6, RA7	Complejidad Computacional	3 semanas

Contenidos	Indicador de logro
5.1. Longitud de una computación. Lenguajes y problemas. Complejidad de un problema. 5.2. Las clases P y NP. Reducción polinomial. NP-completitud. 5.3. El problema NP-completo de satisfactibilidad de fórmulas booleanas. 5.4. Otros problemas NP-completos: clique, recubrimiento de vértices circuito hamiltoniano, coloreo y otros.	La/el estudiante: 1. Analiza y explica el concepto de complejidad computacional y las clases de problemas P, NP, y NP-completos. 2. Identifica y analiza, a partir de un corpus, ejemplos emblemáticos de lenguajes NP-completos como el problema de satisfactibilidad. 3. Demuestra mediante reducciones que ciertos lenguajes son NP-completos. 4. Usa, según corresponda, algún algoritmo de aproximación para resolver un problemas NP-completos. 5. Reporta, en el contexto de las tareas, los resultados de la programación de las soluciones.
Bibliografía de la unidad	[1] Cap. 6 [2] Cap. 6 y 7.1 [3] Cap. 34 [4] Cap. 7

E. Estrategias de enseñanza - aprendizaje:

El curso considera las siguientes estrategias de enseñanza:

- **Clases expositivas** donde se presentan los principales conceptos a trabajar en la sesión, considerando la participación activa de los y las estudiantes y se les estimula, en determinados momentos, a proponer la solución a un problema o desafío planteado.
- **Resolución de problemas,** a partir de tareas en las que se presentan problemas a resolver y entregar en un plazo determinado.

Opcionalmente, se podrán utilizar otras estrategias de aprendizaje, por ejemplo:

Aula invertida (*flipped classroom):* los y las estudiantes deben preparar con antelación su clase, y luego durante la sesión deben resolver problemas guiados por el profesor.

Discusión entre pares: los y las estudiantes analizan y discuten la solución a un problema que se les presenta y luego plantean al resto de la audiencia sus resultados y conclusiones.
El trabajo del y la estudiante es acompañado por el equipo docente, resolviendo dudas cuando se requiera.

F. Estrategias de evaluación:

Al inicio de cada semestre el académico o académica informará al estudiante sobre los tipos de evaluación, cantidad, así como las ponderaciones correspondientes.

Para esta propuesta de programa, el curso considera las siguientes instancias de evaluación:

- **Controles:** resolución de ejercicios simples (entre dos y tres).
- **Tareas teóricas y prácticas (entre dos y tres tareas):** en el contexto de la ejecución de las tareas los y las estudiantes deben describir, analizar e interpretar por escrito los resultados obtenidos a partir de problemas que se les proponen.
- **Examen (1):** busca evaluar, de forma integradora, los aprendizajes adquiridos en el curso.

G. Recursos bibliográficos:

Bibliografía obligatoria:

[1] Navarro, G. (2016). **Teoría de la Computación. Apuntes y Ejercicios**. Disponible en www.dcc.uchile.cl/gnavarro/apunte.html.

Bibliografía complementaria:

[2] Lewis, H, C. Papadimitriou, C. (1998). *Elements of the Theory of Computation*. Prentice-Hall: 2nd edition.
[3] Cormen, T. C. Leiserson, C., R. Rivest, R., Stein, C. (2009). *Introduction to Algorithms.* MIT Press: 3rd edition.
[4] Sipser, M. (2013). *Introduction to the Theory of Computation.* Cengage Learning: 3rd edition.
[5] Kelley, D. (1995). **Teoría de Autómatas y Lenguajes Formales**. Prentice-Hall.
[6] Hopcroft, J., Motwani, R., Ullman, J. (2014). *Introduction to Automata Theory, Languages and Computation.* Pearson Education: 3rd edition.

H. Datos generales sobre elaboración y vigencia del programa de curso:

Vigencia desde:	Primavera, 2021
Elaborado por:	Gonzalo Navarro
Validado por:	Revisión y validación académico par: Alejandro Hevia y Federico Olmedo. Validación de CTD Computación
Revisado por:	Área de Gestión Curricular