TEORIA DE LA COMPUTACION Y VERIFICACIÓN DE PROGRAMAS AVANZADA

Año 2016

Carrera / Plan:
Licenciatura en Informática
Plan 2003/07-2012-2015

Año: 4to
Régimen de Cursada: Semestral

Carácter: Optativa

Correlativas:
Teoría de la Computación y Verificación de Programas
Profesor: Ricardo Rosenfeld
Hs. semanales: 6 hs

FUNDAMENTACIÓN

La materia Teoría de la Computación y Verificación de Programas Avanzada profundiza y extiende los contenidos de la materia correlativa Teoría de la Computación y Verificación de Programas. Fundamentalmente trata la complejidad computacional espacial y la verificación de programas no determinísticos y concurrentes. Por lo tanto, su fundamentación como materia es la misma que la de la materia precedente:
(a) Introduce fundamentos de la teoría de la computación (computabilidad y complejidad computacional) y de la teoría de correctitud de programas (semántica de lenguajes de programación, verificación formal de programas, desarrollo sistemático de programas).
(b) Trata dos importantes pilares de las ciencias de la computación, necesarios en la formación de un profesional de la informática, habiendo éste ya recibido y madurado entre otros, conocimientos de algorítmica y estructuras de datos, matemáticas discretas y lógica matemática.
(c) Como distintos contenidos de la complejidad computacional y de la verificación de programas hoy día están abiertos a distintos caminos de investigación, con esta materia se pretende estimular dicho estudio brindando herramientas básicas y esenciales.

OBJETIVOS GENERALES

Parte 1. Profundización en el estudio de la complejidad computacional espacio-temporal tratada en la materia básica, con foco en la complejidad computacional espacial.

Parte 2. Profundización en el estudio de la teoría de correctitud de programas, con foco en la programación no determinística y concurrente.

CONTENIDOS MÍNIMOS

- ✓ Jerarquía computacional espacial. Espacio logarítmico y polinomial, determinístico y no determinístico.
- ✓ Relación con la jerarquía temporal. Problemas completos de la jerarquía espacio-temporal.
- ✓ Verificación de programas no determinísticos. Fairness.
- ✓ Verificación de programas concurrentes. Modelos de memoria compartida y de pasaje de mensajes. Propiedades de tipo safety y liveness.
- ✓ Introducción a la lógica temporal. Aplicación en la verificación de programas.
- ✓ Introducción a la semántica denotacional.

PROGRAMA ANALÍTICO

Parte 1. Complejidad computacional.

Jerarquía espacial. Espacio logarítmico determinístico y no determinístico. Espacio polinomial determinístico y no determinístico. Teorema de Savitch. Reducciones log-space de problemas. Teorema de Immerman.

Problemas completos de las distintas clases de la jerarquía espacial. Jerarquía espacio-temporal.

Misceláneas: Jerarquía polinomial. Pruebas interactivas. Criptografía. Máquinas cuánticas.

Parte 2. Verificación de programas.

Verificación de programas no determinísticos. Métodos D y D* de verificación de programas no determinísticos. Sensatez y completitud. El concepto de fairness.

Verificación de programas concurrentes con memoria compartida. Métodos de verificación de programas concurrentes con memoria compartida, sin y con primitivas de sincronización (O, O*, R y R*). Sensatez y completitud de los métodos. Propiedades safety y liveness (ausencia de deadlock, exclusión mutua, ausencia de starvation, etc).

Verificación de programas distribuidos (concurrentes sin memoria compartida). Métodos de verificación de programas distribuidos (AFR y AFR*). Sensatez y completitud de los métodos. Propiedades safety y liveness (ausencia de deadlock, exclusión mutua, ausencia de starvation, etc).

Profundización en la verificación de programas secuenciales determinísticos con procedimientos. Recursión y parámetros.

Lógica temporal. Métodos de verificación de programas reactivos basados en la lógica temporal.

Introducción a la semántica denotacional de programas.

METODOLOGÍA DE ENSEÑANZA

La asignatura consiste en el dictado de 15 clases de teoría y 15 clases de ejercitación (clases prácticas), ámbas estrechamente vinculadas y articuladas.

En las clases teóricas se brindan explicaciones conceptuales, con participación e intercambio con los alumnos, imprescindible para el abordaje de los trabajos prácticos.

En las clases prácticas se trabaja a partir del enunciado de ejercicios que se resuelven en las mismas clases, con plena participación de los alumnos.

Para asegurar el aprendizaje de los contenidos dictados, se entrega cada dos semanas un trabajo práctico, a resolver obligatoriamente por los alumnos.

Se utiliza la plataformal virtual de gestión de cursos para la publicación de las clases, trabajos prácticos y artículos de interés. También para las consultas de los alumnos, promoviendo un foro de discusión permanente. En las clases no se proyectan presentaciones, el dictado es con tiza y pizarrón.

También se posibilita la cursada a distancia.

EVALUACIÓN

Se considera directamente la aprobación de la materia, que consiste en la aprobación de los trabajos prácticos quincenales.

Nota: La diversidad y complejidad de algunos temas y su encadenamiento lógico, ameritan que se haga un seguimiento bastante personalizado sobre los alumnos. El mecanismo de trabajos prácticos quincenales ha demostrado ser un buen esquema en este sentido.

BIBLIOGRAFÍA OBLIGATORIA

- ✓ Computabilidad, Complejidad Computacional y Verificación de Programas. Rosenfeld & Irazábal. EDULP 2013.
- ✓ Teoría de la Computación y Verificación de Programas. Rosenfeld & Irazábal. McGraw Hill y EDULP. 2010.
- ✓ Apuntes publicados en la plataforma virtual de gestión de cursos, que varían año a año.

ALGUNA BIBLIOGRAFÍA COMPLEMENTARIA RECOMENDADA

- ✓ Introduction to Automata Theory, Language & Computation. Hopcroft y Ullman. Prentice-Hall. 1979.
- ✓ Computational Complexity. Christos Papadimitriou. Addison-Wesley. 1995.
- ✓ Introduction to the Theory of Complexity. Bovet y Crescenzi. Prentice-Hall. 1994.
- ✓ Computational Complexity: A Conceptual Perspective. O. Goldreich. Cambridge University Press. 2008.
- ✓ Computational Complexity: A Modern Approach. S. Arora y B. Barak. Princeton Univ. 2007.
- ✓ Program Verification. Nissim Francez. Addison-Wesley. 1992.
- ✓ Verification of Sequential and Concurrent Programs. Apt y Olderog. Springer. 1997.
- ✓ Logic in Computer Science. M. Huth y M. Ryan. Cambridge University Press. 2004.
- ✓ The Temporal Logic of Reactive and Concurrent Systems - Specification. Z. Manna y A. Pnueli. Springer-Verlag. 1991.
- ✓ Temporal Verification of Reactive Systems - Safety. Z. Manna y A. Pnueli. Springer-Verlag. 1995.
- ✓ Mathematical Theory of Program Correctness. J. de Bakker. Englewood Cliffs NJ, Prentice-Hall. 1980.

CRONOGRAMA DE CLASES Y EVALUACIONES

Contenidos/Actividades	Evaluaciones
Parte 1. Complejidad computacional. Clase 1. Introducción a la complejidad espacial. Jerarquía espacial. Clase 2. Problemas solubles en espacio logarítimico determinístico y no determinístico. Problemas solubles en espacio polinomial determinístico y no determinístico. Teorema de Savitch. Reducciones log-space de problemas. Teorema de Immerman. Clase 3. Problemas completos de las distintas clases de la jerarquía espacial. Jerarquía espacio-temporal. Clase 4. Misceláneas: jerarquía polinomial, pruebas interactivas, criptografía, máquinas cuánticas. **Parte 2. Verificación de programas.** Clase 5. Verificación de programas no determinísticos. Lenguaje GCL. Sintaxis y semántica operacional. Métodos D y D* de verificación de correctitud parcial y total de programas GCL. Sensatez y completitud de los métodos D y D*.	Trabajos prácticos obligatorios cada 2 clases. Las examinaciones eventuales para la aprobación de la materia se desarrollan mensualmente el primer martes de cada mes.

Clase 6. El concepto de fairness. Fairness débil y fuerte. Verificación de programas no determinísticos asumiendo hipótesis de fairness. Clase 7. Verificación de programas concurrentes con memoria compartida. Lenguaje SVL. Sintaxis y semántica operacional. Métodos de verificación de correctitud parcial y total de programas SVL (O, O*). Sensatez y completitud de los métodos O y O*. Propiedades safety y liveness (ausencia de deadlock, exclusión mutua, ausencia de starvation, etc). Clase 8. Verificación de programas RVL. Sintaxis y semántica operacional. Métodos de verificación de correctitud parcial y total de programas RVL (R, R*). Sensatez y completitud de los métodos R y R*. Propiedades safety y liveness (ausencia de deadlock, exclusión mutua, ausencia de starvation, etc). Clase 9. Verificación de programas concurrentes sin memoria compartida (distribuidos). Lenguaje CSP. Sintaxis y semántica operacional. Método de verificación de correctitud parcial de programas CSP (AFR). Sensatez y completitud del método AFR. Clase 10. Propiedades safety y liveness en el marco de CSP (ausencia de deadlock, exclusión mutua, ausencia de starvation, etc). Método de verificación de correctitud total de programas CSP (AFR*). Sensatez y completitud del método AFR*. Clases 11 y 12. Lógica temporal lineal y computacional. Modelo computacional concurrente. Verificación de programas reactivos utilizando la lógica temporal. Clases 13 y 14. Ampliación de los métodos H y H* considerando la verificación de procedimientos en el lenguaje PLW. Pasaje de parámetros y recursión. Clase 15. Introducción a la semántica denotacional.	

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos)

Direcciones de e-mail del plantel docente:

rrosenfeld@pragmaconsultores.com

Plataforma virtual de gestión de cursos:

Teoría de la Computación y Verificación de Programas. WebUNLP.

Firma del profesor responsable:

Prof. Ricardo Rosenfeld