Una propuesta de integración de nociones lógico-matemáticas en la enseñanza de la Programación

Ariel Ferreira Szpiniak Carlos D. Luna Ricardo H. Medel

Área de Computación
Universidad Nacional de Río Cuarto
{aferreira,cluna,rhmedel}@exa.unrc.edu.ar

Resumen

La formación de profesionales en Informática debe tener como base una formación lógico-matemática muy sólida, que le permita una adecuación rápida y eficaz a los acelerados cambios tecnológicos.

En este artículo se presenta una propuesta de integración de contenidos de la matemática y la lógica en la currícula de computación, inculcando la noción de que los programas son objetos matemáticos plausibles de ser tratados con herramientas lógico-matemáticas.

La propuesta plantea la modificación del desarrollo de la asignatura "Programación Avanzada", en segundo año de las carreras de Analista, Profesorado y Licenciatura en Ciencias de la Computación de la Universidad Nacional de Río Cuarto (UNRC). Se especifican también, los criterios de evaluación de dicha propuesta y algunos resultados parciales obtenidos hasta el presente.

1. Planteo del problema

Los profesionales de la computación deben estar capacitados para estudiar los fundamentos de su disciplina. El núcleo central de las Ciencias de la Computación está constituido en buena parte por la matemática discreta y la lógica matemática (Turner, 1991). En consecuencia, un especialista en computación debe estar en condiciones de usar las herramientas básicas y las técnicas de dichas áreas de la matemática y la lógica.

La necesidad de incorporar conocimientos matemáticos a las carreras de Computación se refleja en la estructura de las currículas de estas carreras en nuestro país. Sin embargo, esta necesidad no siempre es bien comprendida y aplicada por algunas instituciones académicas. Como resultado, existe una corriente de enseñanza de la Computación que prescinde de conocimientos matemáticos fundamentales para esta disciplina, produciendo profesionales con minuciosos conocimientos de la tecnología actual, pero con escasa o nula posibilidad de adaptación a los rápidos cambios que se producen en esta rama de la ciencia.

El desarrollo de habilidades de manipulación matemática debe ser parte de la formación integral de un profesional de la computación. David Gries, en ocasión de recibir el premio anual SIGCSE por su contribución a la educación en Ciencias de la Computación, dejó en claro que tales habilidades son esenciales para poder manipular grandes y complejas estructuras (Gries, 1991). Parnas, un referente fundamental en nuestra disciplina, suscribe este punto de vista: "los programas de Ciencias de la Computación deben retornar al enfoque de la ingeniería clásica dando especial importancia a los fundamentos" (Parnas, 1990).

La enseñanza integrada de matemáticas y programación ha recibido especial atención en la literatura (Henderson and Romero, 1989; Hein, 1993; Wainwrigth, 1992) y numerosos departamentos de computación han adoptado este enfoque (Harrison, 1993). Particularmente, en nuestro país algunas experiencias en este sentido incluyen el desarrollo de las asignaturas "Programación Funcional" en segundo año de las carreras de informática de la UNLP y "Fundamentos para Ciencias de la Computación" en segundo año de la Licenciatura en Cs. de la Computación de la UNS. No sólo agregando asignaturas, sino que basándose en la integración permanente de estas disciplinas es que se enfocó la experiencia de la ESLAI (Escuela Superior LatinoAmericana de Informática), centro de primer nivel de enseñanza de Ciencias de la Computación que brilló en la década del '80, hasta su destrucción en manos de los sicarios de la dependencia.

Lamentablemente, en las carreras de computación de la UNRC, la enseñanza de matemática para los estudiantes de computación se desarrolla como una actividad separada de los conocimientos troncales de la disciplina. En consecuencia, los estudiantes perciben estas habilidades disociadas de la programación.

En las materias específicas de computación, donde se enseñan métodos formales y rigurosos de desarrollo de programas, tales como "Programación Avanzada", "Estructura de la Información" y "Organización de Archivos", resulta evidente que si bien los estudiantes adquieren habilidades de programación y aprenden métodos de especificación, presentan dificultades importantes a la hora de fundamentar rigurosamente y verificar sus programas.

El origen de esta falencia puede atribuirse, entre otras causas, a la corta trayectoria de las carreras de Ciencias de la Computación en esta Universidad. Dichas carreras nacieron en el año 1992, originalmente bajo la responsabilidad del Departamento de Matemática de la Fac. de Cs. Exactas, Fco-Qcas y Naturales hasta la creación del Área de Computación como entidad independiente en el año 1996.

De esta manera, la responsabilidad de los temas impartidos en las asignaturas básicas estuvo a cargo de docentes del Departamento de Matemática, con escasos conocimientos de las Ciencias de la Computación; así las asignaturas de matemática se desarrollaron sin tener en cuenta las aplicaciones que los temas enseñados tienen en computación.

En el Área de Computación de la Facultad de Ciencias Exactas, Fco-Qcas y Naturales de la UNRC, se ha discutido en varias oportunidades esta cuestión. Nuestro punto de vista es que la enseñanza de las matemáticas para Ciencias de la Computación debe ser parte de un curriculum integrado, en el cual se hagan evidentes y se aprovechen las relaciones entre programación y matemáticas.

En este artículo presentamos una propuesta de integración de contenidos de la matemática y la lógica en la enseñanza de la programación, en la asignatura "Programación Avanzada" de las carreras de computación de la UNRC. En la sección 2 damos el marco en el que se desarrolla dicha asignatura hasta la actualidad. En la tercera sección detallamos la propuesta de modificación, tanto de contenidos como de metodología. En la sección 4 se proponen algunos criterios para la evaluación de los resultados obtenidos de la aplicación de la propuesta y, finalmente, en la sección 5 presentamos nuestras conclusiones.

2. Marco de enseñanza

La asignatura "Programación Avanzada" se ubica en el primer cuatrimestre del segundo año de las carreras de Analista en Computación, Profesorado en Ciencias de la Computación y Licenciatura en Ciencias de la Computación de la Universidad Nacional de Río Cuarto.

El curriculum actual comienza en primer año con un curso de programación imperativa ("Diagramación y Programación"). En paralelo se brindan un curso de lógica ("Lógica Matemática Elemental") y uno de álgebra elemental ("Introducción al Álgebra").

En la primera de dichas asignaturas se enseña a programar utilizando esquemas correctos de tratamiento de secuencias, en un lenguaje imperativo semi-formal en castellano, llamado *lenguaje algorítmico* y utilizando la máquina abstracta conocida como Máquina de Caracteres (Lucas, Peyren y Scholl, 1985). Se enseñan también, en forma introductoria, estructuras de datos y recursividad. Las prácticas se realizan implementando los algoritmos en lenguaje Pascal.

En "Lógica Matemática Elemental", por otra parte, se desarrollan temas de lógica proposicional y lógica de predicados, además de nociones de la teoría de conjuntos, distintos tipos de sistemas axiomáticos e inducción matemática.

Así, los alumnos llegan a la asignatura "Programación Avanzada" con elementales conocimientos de programación adquiridos en la única materia de programación previa y con los conocimientos lógicos y matemáticos adquiridos en primer año disociados de conceptos de programación.

El alumnado de esta asignatura está compuesto por estudiantes de las tres carreras de Ciencias de la Computación, por lo que el contenido debe satisfacer las demandas de cada una de las carreras. Este hecho provoca que el programa original de la materia esté conformado por gran variedad de temas sin un hilo conductor, donde cada tema bien podría ser una materia por sí mismo.

En la actualidad la asignatura está a cargo de un equipo conformado por un JTP con dedicación exclusiva, responsable del dictado de las clases teóricas, dos Ayudantes de Primera con dedicación semi-exclusiva, cada uno a cargo de una

comisión de trabajos prácticos, y dos Ayudantes de Segunda Categoría, tal como se sumariza en la Tabla 1.

Este equipo de trabajo es el encargado de realizar la actividad docente que la asignatura requiere, contando en el corriente año (1997) con 123 alumnos.

Cargo	Dedic.	Actividad de formación
JTP	Exc.	Cursando carrera de posgrado
AY1	Semi	Realizando trabajo de grado para Lic.
AY1	Semi	Realizando trabajo de grado para Lic.
AY2	Simple	Cursando 5º año de Lic.
AY2	Simple	Cursando 4º año de Lic.

Tabla 1. Equipo de trabajo de "Programación Avanzada"

La actual conformación del Área de Computación, requiere que el personal docente además de las tareas específicas y de formación personal, realice tareas administrativas por carecer de personal para dichas funciones.

Entre otras limitaciones estructurales podemos mencionar que la bibliografía disponible resulta poco adecuada para esta asignatura y el número de ejemplares es escaso. Por otro lado en la actualidad el laboratorio de informática está conformado por equipos anticuados que sólo alcanzan a cubrir las necesidades básicas requeridas para las prácticas que se necesitan realizar. Esta falencia podría revertirse a partir de la puesta en marcha de un nuevo laboratorio, actualmente en construcción.

Hasta ahora el desarrollo de la asignatura consistía de una clase teórica de 3 horas, de dos clases prácticas de 3 horas cada una y de una clase de consulta de 2 horas. Lo que suma una carga horaria semanal de 11 horas. La modalidad de trabajo de las clases prácticas consistía en la solución de problemas cerrados (Garret, 1988) con más de una solución, requiriendo en algunos casos la realización de trabajos de laboratorio utilizando diversos lenguajes de programación.

3. Propuesta

Nuestra propuesta se basa en articular el desarrollo de la asignatura en tres módulos temáticos principales (ver subsección 3.2.2) y establecer el tratamiento formal de los programas como hilo conductor de los mismos.

3.1 Justificación de la propuesta

La formación de profesionales en Informática para el próximo milenio debe tener como base una formación lógico-matemática muy sólida, que le permita una adecuación rápida y eficaz a los acelerados cambios tecnológicos, que son una constante en la disciplina.

La única garantía de que los futuros profesionales podrán adecuarse a los cambios es que tengan una base teórica suficientemente sólida, combinada con una adecuada experiencia en la aplicación de estos conceptos. Por otra parte, las modernas herramientas de Ingeniería de Software generadas durante los años '90, requieren del conocimiento y la destreza en la utilización de conceptos lógicos y matemáticos cada vez más complejos.

Los estilos declarativos de programación, tales como la programación funcional

(Bird, 1988; Watt, 1990) y la programación lógica (Lloyd, 1993; Watt, 1990), facilitan la integración de la matemática y la programación (Harrison, 1993), dado que permiten utilizar el razonamiento ecuacional para diseñar y optimizar programas. En particular, los programas funcionales son más tratables matemáticamente, permitiéndole al alumno desarrollar sus habilidades para razonar sobre los mismos (Medel, Luna y Ferreira Szpiniak, 1996).

Los conceptos de programación que son parte íntima de los lenguajes funcionales (transparencia referencial, uso de funciones de alto orden como mecanismo de abstracción, sistemas de tipado fuerte con inferencia automática, evaluación perezosa, etc.) brindan un marco teórico sólido, que permite que la enseñanza de las nociones básicas de programación sea sencilla e intuitiva (Baum, Cardós y Martínez López, 1996).

Por otro lado, la programación funcional ha derivado en un disciplina de programación con aplicación en el ámbito del desarrollo de software.

Los cambios de contenidos y de la metodología de enseñanza propuestos para la asignatura en cuestión, apuntan a satisfacer en un nivel básico los mencionados requerimientos. Por una parte, tal como analizamos en (Medel, Luna y Ferreira Szpiniak, 1996), si bien algunos de los temas contenidos en el programa original de "Programación Avanzada" son consistentes con nuestra propuesta, su tratamiento es muy superficial y se les dedica poco tiempo en beneficio de otros temas no centrales en relación con los objetivos propuestos para la asignatura.

En "Programación Avanzada" se pondrá el mayor énfasis en la adquisición de destreza en el uso de herramientas de razonamiento fundamentales: la inducción matemática y la deducción lógica aplicadas a la definición y verificación de los programas. Los objetivos centrales que se persiguen pueden resumirse en la frase "programar rigurosamente sobre la base de argumentos matemáticos", esto es, fortalecer la noción de que junto con la construcción de los algoritmos existe la obligación de la verificación rigurosa (formal) de su corrección y que los programas son objetos matemáticos plausibles de ser tratados con argumentos lógico-matemáticos.

3.2 Detalle de la propuesta

3.2.1 Objetivos

- Presentar a los alumnos la idea de que la programación es una actividad rigurosa, en la que la abstracción juega un rol principal al evitar que el programador se preocupe por las restricciones impuestas por la máquina subyacente (Baum, Cardós y Martínez López, 1996).
- Inculcar a los alumnos la necesidad de discriminar correcta y coherentemente los aspectos más importantes del problema a resolver.
- Destacar la necesidad y utilidad de técnicas formales de diseño de programas y demostración de sus propiedades, como único medio de garantizar la corrección del software producido, promoviendo su utilización y aprovechamiento.
- Dar a conocer a los alumnos la filosofía y los fundamentos del paradigma funcional, en el cual los objetivos anteriores se expresan naturalmente.
- Estimular el desarrollo de habilidades que permitan al alumno resolver problemas sin la asistencia constante del docente, con el objetivo de lograr un grado de independencia tal que le permita en su futuro académico y

profesional enfrentarse y resolver exitosamente nuevos desafíos.

3.2.2 Esquema del programa de la asignatura

El programa propuesto para la asignatura "Programación Avanzada" se divide en tres módulos y un proyecto de implementación. A continuación damos el esquema referenciando las unidades contenidas en cada módulo. En el Anexo A se transcribe el programa completo de la asignatura.

- Introducción a lenguajes formales:
  Unidad 1. Lenguajes formales, gramáticas y autómatas, computabilidad.
- Verificación de programas:
  Unidad 2. Especificación de programas, pre y pos-condición.
  Unidad 3. Técnicas de verificación formal de programas.
  Unidad 4. Una metodología de construcción de programas correctos.
- Programación Funcional:

  Unidad 5. Conceptos de programación funcional moderna.
  Unidad 6. Cálculo lambda.
- Proyecto: Implementación de un sistema de tamaño mediano.

3.2.3 Modalidad de Trabajo

La nueva modalidad de trabajo reduce la cantidad de horas de asistencia a clase sin disminuir las exigencias académicas. El nuevo horario semanal incluye 8 horas entre clases de teoría, trabajos prácticos y consultas. El objetivo de esta reducción es incentivar el trabajo individual de búsqueda bibliográfica, análisis de problemas y su resolución mediante la selección y aplicación de las herramientas conceptuales más adecuadas.

Clases prácticas

El desarrollo semanal de las clases prácticas contempla dos clases cortas de 2 horas cada una, a las cuales se suman dos clases de consulta de 1 hora.

Cada guía de trabajos prácticos contiene problemas de complejidad creciente, incluyendo algunos "problemas abiertos", es decir, con solución desconocida o con varias respuestas adecuadas, cada una para cierto conjunto de circunstancias (Garret, 1988). Se proponen algunos ejercicios de naturaleza netamente teórica, tales como demostraciones y análisis de resultados, y otros de índole práctica. Entre estos últimos se incluyen los que requieren de implementación en lenguajes de programación en el paradigma correspondiente.

Cada ejercicio puede pertenecer a una de tres categorías, a saber: (1) ideados para resolver en clase, (2) adicionales para resolver fuera del horario de clases, con seguimiento mediante las clases de consulta y (3) optativos, ejercicios "interesantes" por su aplicabilidad o integración de contenidos, pero que por complejidad o dimensión, su resolución queda a criterio del alumno.

El énfasis es puesto en ejercicios pertenecientes a la categoría 2, de modo que los alumnos intenten resolverlos sin ayuda del docente, disponiendo del tiempo que a cada uno le resulte necesario y utilizando la bibliografía disponible.

Evaluaciones

Las condiciones para la regularización de la asignatura son: la aprobación de tres exámenes parciales, uno por cada módulo temático, y la presentación de un proyecto

de implementación de un sistema mediano (ver la próxima sección).

Para aprobar la asignatura es necesario rendir un examen final de carácter teórico, involucrando todos los contenidos del programa.

Proyecto

Teniendo en cuenta la importancia que requiere el aprendizaje del paradigma funcional en esta etapa de la carrera (Baum, Cardós y Martínez López, 1996), al finalizar el cuatrimestre se realiza un trabajo que debe ser implementado en un lenguaje funcional puro y perezoso.

Los distintos proyectos son pensados para abarcar reconocimiento sintáctico de lenguajes formales y el uso de diversas características de los lenguajes funcionales, que permitan aplicar las nociones de reutilización de software, programación modular y prototipación rápida (Martínez López y Nestares, 1996). Asimismo, los proyectos incluyen el manejo de conceptos de E/S puramente funcional, que deben ser aprendidos por los alumnos en la última parte de la asignatura.

Un objetivo no menos importante de la realización de estos proyectos es promover el trabajo en grupo, lo que permite desarrollar habilidades de distribución de tareas, desarrollo paralelo y coordinación del trabajo individual de sus integrantes (Aguirre y Arroyo, 1996). Por esto, los alumnos se dividen en grupos de no más de cuatro integrantes, que tendrán la responsabilidad de especificar e implementar el sistema, de confeccionar la documentación técnica y el manual del usuario, y de realizar una exposición individual en defensa del trabajo.

Algunos de los proyectos propuestos son:

- Realización de un derivador de expresiones matemáticas, incluyendo la confección de un reconocedor, un simplificador y un evaluador de expresiones.

- Desarrollo de un Sistema de Cruzamiento de Encuestas que dado un archivo de datos numéricos conteniendo los resultados de una encuesta y un archivo de texto describiendo una tabla de cruzamiento (en un cierto lenguaje), imprime un cuadro de frecuencias de aparición de los datos con el formato de la tabla (Aguirre y Medel, 1997).

- Desarrollo de una Máquina de dibujar, que dado un programa como una serie de instrucciones similares a las órdenes LOGO, ejecute dicho programa y retorne los puntos dibujados. Estos datos son enviados a un programa imperativo que imprime el dibujo.

- Construcción de un programa funcional interactivo que permita actualizar y consultar (en forma eficiente) términos en un diccionario, utilizando un lenguaje de consultas predefinido.

Desarrollar un sistema que permita crear y utilizar planillas de cálculo, en forma simplificada.

3.2.4 Determinación de los roles de cada miembro del equipo docente

Los cinco docentes afectados a la planificación y desarrollo de la asignatura realizarán las siguientes tareas:

- Reunión semanal para:
  Analizar y discutir la aplicación de la metodología propuesta y posibles mejoras aplicables en clase.
  Mantener la coordinación del cronograma de clases teóricas con las

clases prácticas.

- Confección de guías de trabajos prácticos y discusión de la forma de enfrentar a los alumnos con los problemas propuestos.
- Confección del manual con los contenidos de la asignatura, destinado a servir de bibliografía base para los alumnos.
- Elaboración y corrección de exámenes parciales y recuperatorios.
- Desarrollo de una base de datos conteniendo problemas factibles de ser utilizados como ejercicios en clases prácticas o exámenes parciales.
- Elaboración de los proyectos de implementación propuestos y seguimiento de los grupos de trabajo.

4. Forma de evaluación de la propuesta

La evaluación de los resultados de esta propuesta se realizará en base a los siguientes criterios:

- Tipo de preguntas realizadas por los alumnos al docente con motivo de la resolución de los problemas planteados, en clases prácticas y de consulta,
- Nivel alcanzado por los alumnos en la resolución de los problemas planteados en los exámenes parciales y finales,
- Tasa de ocurrencia de los errores típicos en programación funcional, listados en (Clack and Myers, 1995),
- Calidad de los sistemas informáticos resultantes de los proyectos de implementación llevados a cabo. Específicamente en cuanto a sus características de corrección, nivel de modularización, correcta aplicación de las características del lenguaje de programación utilizado y calidad de la documentación,
- Resultados obtenidos por los alumnos en las asignaturas posteriores que involucren la resolución de problemas y el desarrollo formal de programas.

5. Conclusiones y trabajo futuro

Bajo la premisa de que sólo una formación lógico-matemática muy sólida permitirá a los profesionales de computación una adecuación rápida y eficaz a los acelerados cambios tecnológicos, consideramos que el curriculum de las carreras de Ciencias de la Computación debe hacer evidentes y aprovechar las relaciones entre programación y matemática.

El objetivo de esta propuesta es incorporar, en una asignatura temprana de la carrera, conceptos fundamentales en dicha integración y permitir una adecuada experiencia en la aplicación de las herramientas de razonamiento formal.

Los tres módulos que conforman la asignatura apuntan a fortalecer la noción de que junto con la construcción de los algoritmos existe la obligación de la verificación formal de su corrección, y que los programas son objetos matemáticos plausibles de ser tratados con argumentos lógico-matemáticos.

En la actualidad, esta propuesta está siendo implementada como parte de un Proyecto Pedagógico Innovador dentro del Área de Computación de la UNRC. Si bien aún no se ha realizado la evaluación final de los resultados obtenidos en el primer año, podemos mencionar que los resultados parciales muestran un considerable

aumento de la participación activa de los alumnos en proponer diferentes soluciones a los problemas presentados en las clases prácticas.

Asimismo, la cantidad de alumnos en condiciones de regularizar se ha incrementado notoriamente respecto de años anteriores, lo que hace presuponer una mejora en la comprensión de los temas involucrados en la asignatura.

Como contraparte, es preciso mencionar que a pesar de que se han aumentado los requerimientos de estudio autoasistido y búsqueda bibliográfica, los resultados en ese aspecto no muestran una variación significativa.

Debe quedar en claro que, independientemente de los resultados puntuales obtenidos en esta asignatura, el cumplimiento del objetivo primordial de esta propuesta, esto es, la incorporación de habilidades matemáticas y su integración con los conceptos de programación, sólo podrá evaluarse en un contexto global, teniendo en cuenta el rendimiento de los alumnos en las demás asignaturas del plan de estudios de cada carrera.

Es preciso mencionar que la consecución de los objetivos formulados se basa en la cohesión de los contenidos de esta asignatura con las anteriores y las posteriores, puesto que la integración buscada no podrá lograrse sólo mediante la modificación de una asignatura en forma aislada.

En este sentido, está en estudio la implementación de un Proyecto Pedagógico Innovador conjunto con la asignatura "Estructura de la Información", en la que se desarrollan los temas de estructuras de datos, Tipos Abstractos de Datos, especificaciones algebraicas de TAD y nociones de complejidad de algoritmos.

Además, se está llevando adelante la modificación de los planes de estudio de las tres carreras, en función de lograr su actualización tomando como eje principal los mismos objetivos planteados en esta propuesta.

Bibliografía

Aguirre, J. y Arroyo, M. (1996) "*Una experiencia en la enseñanza de teoría de lenguajes y compiladores*", 2º Congreso Argentino de Ciencias de la Computación, 4º Ateneo de Profesores Universitarios de Computación y 3º Workshop sobre Aspectos Teóricos de Inteligencia Artificial, 7 al 9 de Noviembre de 1996, Universidad Nacional de San Luis, pp. 507-519.

Aguirre, J. y Medel R. (1997) "*Un proyecto de construcción de software de aplicación para ser desarrollado en los primeros años de una carrera de Ciencias de Computación: Un sistema de cruzamiento de encuestas*", artículo en preparación.

Baum, G. A., Cardós, M. y Martínez López, P. E. (1996) "*Programación Funcional en la Enseñanza de Grado: Motivaciones y Experiencias*", Anales del 1º Taller de Programación Funcional, 25º Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, 9 al 12 de Setiembre de 1996, pp. 107-115.

Bird, R. and Wadler, P. (1988) "*Introduction to Functional Programming*", Prentice Hall International, Hemel Hempstead, England.
Clack, Ch. and Myers, C. (1995) "*The Dys-Functional Student*" en Hartel, P.H. and Plasmeijer, R. (editores) "*Functional Programming Languages in Education*", Proceedings of the First International Symposium, FPLE'95, Nijmegen, The Netherlands, 4 al 6 de Diciembre de 1995, LNCS 1022, Springer.

Garret, R. M. (1988) "*Resolución de Problemas y Creatividad: Implicaciones para el Currículo de Ciencias*", Enseñanza de las Ciencias, 6(3):224-230.

Gries, D. (1991) "*Improving the curriculum through the teaching of calculation and discrimination*", Education and computing, 7(1,2).

Harrison, R.(1993) "*The use of functional programming languages in teaching computer science*", Journal of functional programming, 3(1):67-75.

Hein, J.L. (1989) "*A declarative laboratory approach for discrete structures, logic and computability*", ACM SIGCSE bulletin, 25(3):19-24.

Henderson, P.B. and Romero, F.J. (1989) "*Teaching recursion as a problem-solving tool using standard ML*" in Barrett, R.A. and Mansfield, M.J. (editors), "*20th Computer science education*", ACM SIGCSE bulletin, 21(1):27-31.

Lucas, M., Peyren, J.-P. y Scholl, P.-C. (1985) "*Algorítmica y representación de datos, Tomo 1: Secuencias, Autómatas de estados finitos*", Editorial Masson, Barcelona.

Lloyd, J. W. (1993) "*Foundation of Logic Programming*", Springer Verlag, New York.

Martínez López, P. y Nestares, G. (1996) "*Parsers Funcionales Genéricos*", 2º Congreso Argentino de Ciencias de la Computación, 4º Ateneo de Profesores Universitarios de Computación y 3º Workshop sobre Aspectos Teóricos de Inteligencia Artificial, 7 al 9 de Noviembre de 1996, Universidad Nacional de San Luis, pp. 437-449.

Medel, R., Luna, C. y Ferreira Szpiniak, A. (1996) "*Experiencias en la Enseñanza de Programación Funcional en las carreras de Ciencias de la Computación de la Universidad Nacional de Río Cuarto*", Anales del 1º Taller de Programación Funcional, 25º Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, 9 al 12 de Setiembre de 1996, pp. 95-106.

Parnas, D.L. (1990) "*Education for computing professionals*", Computer, 23(1):17-22.

Turner, A.J. (1991) "*A summary of the ACM/IEEE-CS joint curriculum task force report: Computing curricula 1991*", Communications of ACM, 34(6):69-84.

Wainwrigth, R.L. (1992) "*Introducing functional programming in discrete mathematics*", in Mansfield, M.J., White, C.M. and Hartman, J. (editors), "*23rd Computer science education*", ACM SIGCSE bulletin, 24(1):147-152.

Watt, D. A. (1990) "*Programming Language Concepts and Paradigms*", Prentice Hall International, Hemel Hempstead, England.

Anexo A

Programa de la asignatura "Programación Avanzada"

Unidad I: Gramáticas, Autómatas y Computabilidad.

Lenguajes. Clasificación según Chomsky. Gramáticas. Formalismo de Backus Naur

(BNF). Construcción de árboles de derivación. Gramáticas regulares. Expresiones regulares. Autómatas finitos. Incorporación de significado mediante el agregado de acciones a los arcos. Otros modelos de autómatas. Máquina de Turing. Introducción a la computabilidad. Demostración informal de que determinar si un procedimiento es un algoritmo no es decidible.

Unidad II: Especificaciones funcionales.

Especificación funcional. Pre y pos-condición, especificación de las máquinas abstractas usadas para modelizar el tratamiento de secuencias. Especificación de una "máquina de dibujar" tipo "dibujos de tortuga". Procedimientos recursivos. Dibujos recursivos.

Unidad III: Verificación de programas.

Verificación de programas. Corrección parcial y total. Definición funcional de estado. Expresiones atómicas. Predicados. Sustitución textual. Definición funcional de arreglos. Pictures (dibujos) para representar propiedades sobre arreglos. Uso de aserciones para documentar programas. Bosquejo de demostración. El transformador de predicados *wp* (precondición más débil). Definición, mediante el *wp*, de los constructores del lenguaje SMALL. Constructores SKIP, ABORT, secuencia (;) y asignación (:=). El constructor alternativo IF. Teorema del IF. El constructor iterativo DO. Definición formal. Predicado invariante y función acotada. Teorema del DO. Lista de verificación (Checklist) de un ciclo.

Unidad IV: Construcción de programas correctos.

La programación como una actividad dirigida por metas. Construcción de programas a partir de sus poscondiciones. Construcción de ciclos a partir de su invariante. Construcción de invariantes. Estragegias: supresión de un conjuntor, extensión del rango de una variable, reemplazo de una constante por una variable y combinación de pre- y poscondiciones.

Unidad V: Programación Funcional.

Programación Funcional moderna. Expresiones de tipos. Definición de tipos. Inferencia de tipos. Sobrecarga de funciones y coerción. Evaluación perezosa. Uso de objetos infinitos. Funciones de orden superior. Tipos algebraicos de datos. Entrada/Salida puramente funcional. Demostración de programas funcionales. Utilización del lenguaje Gofer para implementar soluciones a problemas utilizando los conceptos vistos.

Unidad VI: Cálculo Lambda.

El cálculo lambda. Expresiones. Variables libres y ligadas. Sustituciones. Transformaciones. Orden de reducción. Forma Normal. Enunciado del Teorema de Church-Rosser y su corolario: Unicidad de la Forma Normal.

Condiciones para regularizar:

1. Aprobar tres exámenes parciales, cada uno con una posibilidad de recuperación.

2. Realizar un Proyecto de implementación utilizando el paradigma funcional, con el fin de integrar los temas estudiados en la materia y en asignaturas previas.