Errores que cometen los alumnos de nuestro curso de Programación Funcional

Ricardo H. Medel, María Marta Novaira[1], Albino Scoppa

Área de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto
{rhmedel, mnovaira, ascoppa}@exa.unrc.edu.ar

Resumen

En el presente trabajo se reportan los resultados preliminares de analizar la ocurrencia de errores de programación en el paradigma funcional, cometidos por los alumnos de las carreras de Cs. de la Computación de la UNRC en los exámenes parciales de la asignatura "Programación Avanzada" durante los años 1996 y 1997. Se reconocen y clasifican los errores de programación en el paradigma funcional y se proponen mejoras en el dictado de la asignatura a fin de minimizar la aparición de dichos errores.

1. Introducción

Hay un amplio consenso en que el desarrollo de habilidades de uso y manipulación de herramientas de la matemática y la lógica debe ser parte fundamental en la enseñanza de la programación para los futuros profesionales de Ciencias de la Computación. Los estilos de programación declarativos, como la programación funcional, facilitan la integración de los contenidos matemáticos con los específicos de programación, ya que permiten inculcar la noción de que los programas son objetos matemáticos plausibles de ser tratados con herramientas lógico-matemáticas [BW88].

Esta ventaja es aprovechada por cada vez más instituciones educativas que aplican la enseñanza de la programación funcional en sus carreras de grado en Cs. de la Computación, incluyéndola en cursos de resolución de problemas, algoritmos, estructuras de datos, compiladores, arquitectura del procesador, computación gráfica y semántica de lenguajes de programación [HP95].

Considerando trabajos previos [BCM95][HET95][CM95], que analizan las características de la enseñanza de la programación funcional con el fin de dar valiosas sugerencias sobre cómo evitar errores en el dictado de asignaturas que contengan dicho tema, se comenzó a trabajar en un proyecto que nos permitiera mejorar la eficacia en la enseñanza de la programación funcional en las carreras de Cs. de la Computación que se dictan en esta Universidad, aprovechando la experiencia obtenida durante el dictado de la asignatura "Programación Avanzada".

Del análisis de las experiencias vividas desde 1993 en el dictado de la asignatura, surgieron varias propuestas de modificación del programa. Estas propuestas fueron publicadas en [MLF96], [FLM97] y [FLM98]. A fin de estudiar los efectos de la implementación de dichas modificaciones, se consideró analizar las diferencias encontradas en los errores de programación producidos por los alumnos durante los distintos años de dictado de la asignatura.

En el presente trabajo se reportan los resultados parciales obtenidos de analizar los errores de programación en el paradigma funcional cometidos por los alumnos de "Programación Avanzada" durante los años 1996 y 1997.

La segunda sección explica el método de análisis de errores seguido en el trabajo. La tercera sección discrimina los distintos errores encontrados. La cuarta sección está dedicada a mostrar la cantidad de ocurrencias de los mencionados errores. En la quinta sección se analizan las causas probables de la aparición de los errors mencionados y en

[1] Trabajo realizado en el marco de la Ayudantía de Investigación: "*Análisis de correlación de la ocurrencia de errores de programación en el paradigma funcional*", director Lic. Gabriel Baum.

la sexta y última sección se exponen las conclusiones y posibles extensiones al presente trabajo.

2. Casos de estudio

En el caso de las carreras de Computación de esta Universidad, la programación funcional se enseña como tal en "Programación Avanzada" (código 1948), asignatura que se comenzó a dictar en el año 1993 y es compartida por las carreras de Analista en Computación, Profesorado en Cs. de la Computación y Licenciatura en Cs. de la Computación. Sin embargo, sus conceptos son utilizados y revisados en asignaturas posteriores, tales como en "Estructura de la Información" (1954), "Organización de Archivos" (1955) y "Análisis Comparativo de Lenguajes" (1956).

En el presente trabajo se analizaron los errores de programación cometidos por los alumnos de la asignatura "Programación Avanzada" en los ejercicios de los exámenes parciales que incluyeran tópicos de programación funcional. La idea original era observar los cambios ocurridos en la frecuencia de aparición de los errores durante los años en que la asignatura fue dictada, a fin de analizar el efecto de las modificaciones en el dictado de la materia propuestas en [MLF96], [FLM97] y [FLM98].

Lamentablemente, al comenzar el trabajo se descubrió que los exámenes parciales de los años 1993 a 1995 no fueron archivados. Ante esta falta de la documentación necesaria para el trabajo, se debió reorientar el proyecto para analizar los exámenes disponibles, esto es, de los años 1996 y 1997. Mientras se analizaban los datos mencionados, la asignatura se dictó en el año 1998, incluyendo el desarrollo de un proyecto de implementación en el paradigma funcional, y se encuentra en pleno dictado del año 1999.

Por esta razón los datos incluidos en el presente reporte sólo abarcan los años 1996 y 1997, los cuales, a pesar de ser una muestra relativamente pequeña respecto de toda la experiencia obtenida durante el dictado de la asignatura, resultan ser significativos como base de partida para el reconocimiento y clasificación de los errores de programación en el paradigma funcional.

Dado que el tema de la programación funcional es sólo una parte de la asignatura "Programación Avanzada", se analizaron sólo los ejercicios que correspondían a este tema. En total se revisaron 1.148 ejercicios. En la siguiente tabla se indica qué ejercicios de qué parciales de cada año fueron revisados.

Año	**Examen parcial**	**Ejercicios**	**Cant. alumnos**
1996	2° parcial	1, 2, 3, 4	72
	Recuperatorio 2° parcial	1, 2, 3, 4	27
1997	3° parcial	1a, 1b, 3ª, 3b, 3c, 4	74
	Recuperatorio 3° parcial	1, 2a, 2b, 3, 4a, 4b, 4c	44

Tabla 1. Ejercicios analizados.

Es deseable que este trabajo se continúe con el análisis de los resultados de los exámenes producidos durante los años de dictado subsiguientes, especialmente el proyecto de implementación realizado en 1998, y la verificación empírica de los cambios producidos por la aplicación de las mejoras sugeridas en el presente artículo.

3. Errores comunes de Programación Funcional

En base a los errores de programación en el paradigma funcional reportados en el trabajo de Clack y Myers [CM95], se confeccionó una lista tentativa de errores de posible ocurrencia en los programas funcionales realizados por los alumnos.

Del análisis de los resultados obtenidos en la corrección de los ejercicios mencionados en la Tabla 1, resultó claro que el listado confeccionado previamente no

cubría adecuadamente todos los casos hallados. Además, como posteriormente se verá, algunos de los errores reportados en [CM95] no aparecieron en los ejercicios analizados.

Se propuso, entonces, una nueva lista de treinta y seis (36) errores de programación en base a la lista anterior y a los nuevos errores detectados en la revisión de los parciales mencionados. A continuación, la Tabla 2 muestra la lista de los errores cuya aparición en los ejercicios fue controlada durante este trabajo.

Errores comunes de Programación Funcional

1. Utilizar el tipo **string** para mostrar un resultado lógico (**Bool**).
2. Testeos innecesarios.
3. Falta la **declaración del tipo** de la función, antes de la definición de la misma.
4. No advierte algún **caso base**.
5. Caso base demás.
6. Caso base demasiado general.
7. Confundir el tipo **entero** (**Int**) con los **naturales positivos**.
8. Confundir **cons** (**:**) con **concat** (++), o no utiliza bien el **cons** (**:**).
9. Confundir **[]** con **[[]]**.
10. Error en **pasaje de parámetros** en invocación de funciones.
11. Parametrización (Curry).
12. En el pattern sobran o faltan parámetros.
13. No usa **funciones de alto orden**.
14. Redefine **caso base** cuando utiliza una función de **alto orden**.
15. **No utiliza bien** la función de alto orden.
16. Ausencia de **comentarios**.
17. Utiliza demasiadas sentencias **error**.
18. Utiliza demasiadas sentencias **where**.
19. No usa bien **pattern matching**.
20. Mal el **tipo** de la función.
21. Instancia mal funciones conocidas.
22. Mal **asumción**.
23. Mal notación de las listas infinitas.
24. Error en **notación** de Gofer.
25. Mal orden en **invocaciones de funciones**.
26. No se corresponde el **resultado de la función con el tipo definido** de la misma.
27. Confundir **data** con **type** y **class**.
28. Utilización de un operador no instanciado.
29. Falta definir **data** o **type**.
30. Mal definido un nuevo tipo de datos.
31. Mal notación de listas por comprensión.
32. Problemas con expresiones lambda.
33. Usa un caso base demás por no utilizar el pattern usual de lista.
34. Error en la llamada recursiva.
35. Refiere mal los elementos de un tipo estructurado.
36. Utiliza identificadores de tipo como variables.

Tabla 2. Listado de errores de programación.

Por cuestiones de espacio, en este artículo sólo listamos los nombres dados a cada error, en un Reporte Técnico [MNS], resultado de nuestro trabajo de investigación, se da la lista de todo los errores con su correspondiente explicación y un ejemplo aclaratorio.

Para simplificar el análisis, estos errores fueron clasificados en siete categorías, las cuales se muestran en la siguiente tabla:

Clase	**N° identificador de errores**
Tipos	1, 3, 7, 10, 11, 12, 20, 21, 26, 27, 28, 29, 30, 35, 36
Listas	8, 9, 23, 31, 33
Notación	19, 23, 24, 31, 32, 36
Orden Superior	11, 13, 14, 15, 25
Recursividad	4, 5, 6, 14, 34
Habilidades básicas de programación	2, 16, 17, 18, 22
Cálculo Lambda	32

Tabla 3. Clasificación de errores de programación.

4. Frecuencia de aparición de errores

A continuación se muestra para cada categoría de errores, la cantidad de veces que un error perteneciente a dicha categoría aparece en la solución de un ejercicio, discriminado por el examen parcial que corresponde.

Tipo	Parc96	254
	Rec96	54
	Parc97	320
	Rec97	191

Listas	Parc96	50
	Rec96	21
	Parc97	38
	Rec97	100

Notación	Parc96	59
	Rec96	10
	Parc97	27
	Rec97	71

Orden Superior	Parc96	95
	Rec96	19
	Parc97	65
	Rec97	31

Recursividad	Parc96	28
	Rec96	28
	Parc97	69
	Rec97	12

Hab. Básicas de Programación	Parc96	30
	Rec96	0
	Parc97	1
	Rec97	1

Cálculo Lambda	Parc96	0
	Rec96	2
	Parc97	0
	Rec97	0

Tabla 4. Ocurrencia de errores por categoría y por parcial.

Sumando los totales, se puede obtener el siguiente gráfico de ocurrencia de errores en todos los ejercicios revisados durante nuestro trabajo.

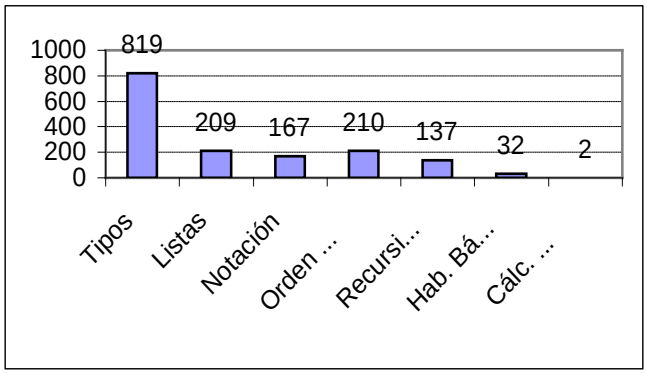

Gráfico 1. Ocurrencia de errores por categoría.

5. Análisis de errores ocurridos

Analizando los datos mostrados en el gráfico 1 de la sección anterior, observamos una notable superioridad de la cantidad de ocurrencias de errores de "tipos" por sobre cualquier otra categoría de error. Esto puede deberse a la cantidad de errores de esta categoría, desde el simple "falta la declaración del tipo de la función" hasta el de "Parametrización (Curry)" que implica no comprender o no utilizar la *currificación*.

Por otro lado, el sistema de tipos Hindley-Milner es muy poderoso pero probablemente complicado de aprender para los alumnos que sólo han cursado previamente un curso introductorio de algorítmica basado en un lenguaje imperativo.

Las categorías siguientes en cantidad de errores se refieren a errores de listas y de orden superior. Justamente estos temas son totalmente nuevos para los alumnos y son introducidos en esta asignatura. El amplio uso de la listas en nuestros ejercicios, casi como única estructura de datos disponible, hace que la cantidad de errores de este tipo se multiplique.

Significativamente, las categorías de errores de "Notación" y "Recursividad" no aparecen con mayores ocurrencias. La primera de estas categorías puede beneficiarse de la utilización del lenguaje Gofer, cuya notación es muy simple y similar a la matemática. Para la segunda categoría, la explicación bien puede residir en que el tema de recursividad es estudiado por los alumnos desde fines del primer año, y luego reforzado al comenzar la asignatura "Programación Avanzada", aunque en el paradigma imperativo.

Los errores pertenecientes a la categoría "Habilidades Básicas de Programación", están basados observados por Clack y Myers [CM95]. Sin embargo, tienen una frecuencia muy baja de aparición entre nuestros alumnos. Esto sugiere que los errores de programación mencionados están basados en una mala formación en algorítmica.

Mencionamos como ejemplo de esta situación al error 1: "*Utilizar el tipo **string** para mostrar un resultado lógico (**Bool**)*". Este error es mencionado como frecuente por Clack y Myers, mientras que entre nuestros alumnos tal error nunca apareció. Parece claro que este error ocurre en los alumnos de los autores citados debido a que dichos alumnos se encuentran en el último año de su carrera de grado, que tienen una fuerte formación en programación imperativa basada en lenguajes que no tienen el tipo lógico entre sus tipos primitivos. Nuestros alumnos, en cambio, vienen de sólo un curso de introducción a la algorítmica donde tanto la notación algorítmica (o pseudo-código) como el lenguaje de práctica (Pascal) incluyen al tipo lógico entre los primitivos.

Finalmente, cabe aclarar que la ocurrencia ínfima de la categoría "Cálculo Lambda", además de ser consecuencia de contener un único error, también es producto de la poca cantidad de ejercicios en los exámenes que evalúan el uso de esta notación, ya que usualmente es un tema dado al final del cuatrimestre y por lo general no se incluye como tema de parcial.

6. Conclusiones y trabajo futuro

Del análisis precedente podemos concluir que el sistema de tipos de los lenguajes funcionales modernos, como Gofer o Haskell, presenta una dificultad mayor para los alumnos que por primera vez se exponen al paradigma funcional. Por esto, cabe sugerir que durante el cursado de la asignatura se ponga énfasis en la enseñanza de este tema.

En cuanto a las otras categorías de errores, se concluye que la cantidad de ocurrencias es la esperable. En particular, el lenguaje Gofer ha permitido que la notación se mantenga simple y a la vez puedan escribirse algoritmos poderosos, por lo que debería seguir utilizándose éste u otro lenguaje similar, tal como Haskell.

Respecto de la baja ocurrencia de errores de la categoría "Recursividad" indica que este tema está bien incorporado por los alumnos, por lo que debe seguir dictándose al

final del primer curso introductorio a la algorítmica y luego reforzarse durante las etapas *imperativas* de la asignatura "Programación Avanzada".

Por último, es destacable la virtual inexistencia de errores producidos como resultado de aplicar técnicas imperativas en el paradigma funcional. Esto es producto de la introducción del paradigma en los primeros años de la carrera, evitando que los alumnos caigan en las *garras del imperativo* [CM95]. Entonces, se recomienda mantener el tema de la programación funcional en el primer cuatrimestre de segundo año o incluso exponer a los alumnos a este paradigma aún en cursos inferiores.

Como mencionamos, este artículo es un reporte preliminar de los resultados de nuestro trabajo. Por lo que aún se debe realizar un análisis por error, a fin de encontrar patrones de ocurrencias cuyo análisis permita mejorar el dictado de la materia.

Por otra parte, quedan por evaluar el dictado de la asignatura en los años 98 y 99, en particular el proyecto de implementación realizado durante el primero de ellos.

Los resultados finales de este trabajo serán publicados como Reporte Técnico del Área de Computación, a fin de que estén disponibles para todos los docentes que lo requieran.

Referencias

[BCM95] Baum, G., Cardós, M. y Martínez López, P.E., "Programación Funcional en la Enseñanza de Grado: Motivaciones y Experiencias", 1° Taller de Programación Funcional, 25° JAIIO, Buenos Aires, Setiembre de 1996, págs. 107-115.

[BW88] Bird, R. and Wadler, P., "Introduction to Functional Programming", Prentice Hall, 1988.

[CM95] Clack, C. and Myers, C., "The Dys-functional student", en [HP95], págs. 289-309.

[FLM97] Ferreira Szpiniak, A., Luna, C.D. y Medel, R.H., "Una propuesta de integración de nociones lógico-matemáticas en la enseñanza de la Programación", CACIC'97, La Plata, Setiembre-Octubre de 1997, págs. 881-892.

[FLM98] Ferreira Szpiniak, A., Luna, C.D. y Medel, R.H., "Our Experiences Teaching Functional Programming at University of Río Cuarto (Argentina)", SIGCSE Bulletin, Vol.30, N°2, June 1998, págs. 28-30.

[HET95] Hartel, P.H., van Es, B. and Tromp, D., "Basic proof skills of computer science students", en [HP95], págs. 269-287.

[HP95] Hartel, P.H. and Plasmeijer, R., "Functional Programming Languages in Education", Proceedings of First International Symposium FPLE'95, Nijmegen, The Netherlands, December 1995, LNCS 1022, Springer-Verlag.

[MLF96] Medel, R.H., Luna, C.D. y Ferreira Szpiniak, A., "Experiencias en la Enseñanza de Programación Funcional en las carreras de Ciencias de la Computación de la Universidad Nacional de Río Cuarto", 1° Taller de Programación Funcional, 25° JAIIO, Buenos Aires, Setiembre de 1996, págs. 95-106.

[MNS99] Medel, R.H., Novaira, M.M. y Scoppa, A., "Reporte Técnico: Errores de Programación Funcional encontrados en alumnos de Programación Avanzada en el período 1996-1997", Área de Computación, Fac. de Cs. Exactas, Fco-Qcas y Naturales, UNRC, *en preparación*.