Formación en Matemática Discreta usando Lenguajes de Programación

Sylvia Rita da Rosa
Universidad de la República
Instituto de Computación,
Montevideo, Uruguay, 11300
darosa@fing.edu.uy

and

Gustavo Anibal Cirigliano
Universitario Autónomo del Sur,
Depto. Teoría de la Computación,
Montevideo, Uruguay, 11100
gcirigli@ei.edu.uy

Abstract

We have been teaching introductory courses on Functional Programming, as part of the curricula of computer science University studies, for several years. Through our experience we arrived at the conclusion that the origin of most of the difficulties in learning Functional Programming doesn't lie in computer science courses but in the mathematical background of the students: for instance Discret Mathematics. Of course, this means that the problem is much harder to solve, because it is concerned with changes involving not only computer science education but also mathematics education. On the other hand, we are convinced that an integrated work of mathematics and computer science educators will considerably benefit the learning of both subjects.Our proposal mainly consists in formulating a new approach to teach discrete mathematics topics included in High School mathematics curricula, relating them with computer science concepts. This implies, on one hand, to increase the dedication to discrete mathematics in High School curricula, and on the other hand, to introduce a programming language as a formalism to manipulate the objects. We put our proposal into practice, teaching a discrete mathematics course using a programming language (ISETL) to a group of High School mathematics teachers, with encouraging results summarized in the conclusions.

Keywords: Discret Mathematics, Tools for teaching, ISETL, Functinal Programming.

Resumen

Desde hace algunos años, nuestra actividad como docentes se centra en cursos introductorios de Programación Funcional en carreras universitarias de Ciencia de la Computación. Nuestra experiencia nos ha permitido constatar que la mayoría de las dificultades en el aprendizaje de Programación Funcional, no se encuentren en los cursos de computación, sino en la formación matemática básica de los estudiantes, por ejemplo Matemática Discreta. Por supuesto, esto significa que el problema es más difícil de resolver, porque involucra cambios no sólo en la enseñanza de computación sino también en la de matemática. Nuestra propuesta consiste fundamentalmente en la formulación de un enfoque nuevo para la enseñanza de tópicos de matemática discreta incluidos en el curriculum de los cursos de Secundaria, que los relacione con conceptos de ciencia de la computación. Esto implica, por un lado, el aumento de la dedicación a matemática discreta en la currícula de Secundaria y por otro, la introducción de un lenguaje de programación como un formalismo para manipular los objetos matemáticos. Hemos puesto nuestra propuesta en práctica, dictando un curso de matemática discreta usando el lenguaje de programación ISETL a un grupo de docentes de matemática de Secundaria obteniendo resultados prometedores, como se describe en las conclusiones de este trabajo.

Palabras claves: Matemática Discreta, Herramientas informáticas en la educación, IsetL, Programación Funcional

1 Introducción

Hemos detectado que los estudiantes que ingresan a la universidad, tienen serias dificultades para comprender conceptos como funciones de alto orden, recursión, inducción, sistemas de tipos y lógica en general. Por otro lado, esos conceptos son ampliamente *usados* por los estudiantes para adquirir destrezas de cálculo: trabajan con derivadas (funciones de alto orden), usan conjuntos como dominio y recorrido de funciones (noción similar a sistemas de tipos), usan algoritmos recursivos, (Euclides, Ruffini, etc), y hacen pruebas usando inducción matemática (inducción y lógica).

Sin embargo, las dificultades en aprender los conceptos mencionados, no son exclusivas de la enseñanza de ciencia de la computación, por el contrario, son también señaladas por docentes de matemática. Para mejorar esta situación, se vienen realizando varios esfuerzos: del lado de la enseñanza de matemática, se introducen herramientas tecnológicas para ilustrar ideas, para producir resultados eficientes (cálculos, grafos, figuras en 3 dimensiones, etc.) y del lado de ciencia de la computación, la preocupación acerca de los problemas de aprendizaje se ha incrementado considerablemente [1][2][10]. Entre las distintas metodologías propuestas y puestas en práctica, mencionamos las que ponen el acento en comenzar los estudios en computación aprendiendo a programar en un lenguaje declarativo. Uno de los primeros en promover esta idea, usando un lenguaje funcional, fue el MIT en 1980. Respecto a los lenguajes imperativos, resaltaban las siguientes ventajas:

a) el uso de un paradigma diferente al utilizado en el campo profesional y al que puedan haber utilizado en la enseñanza media, permite el abordaje de los temas a un nivel conceptual mayor, y reduce la heterogeneidad de conceptos previos entre los estudiantes.

b) la sintaxis relativamente sencilla permite concentrarse en los aspectos conceptuales de la teoría.

c) algunas ideas importantes (recursión, listas, funciones, etc) surgen naturalmente en estos lenguajes.

También alertaban sobre algunos peligros:

a) los estudiantes reaccionan escépticamente a lenguajes que no están entre los utilizados en el mercado, o los que puedan tener en mente.

b) Se requiere que los estudiantes trabajen a un nivel más abstracto que el utilizado en los lenguajes imperativos.

Nosotros mismos experimentamos este estilo, introduciendo Haskell en un curso de Matemática Discreta en 1997 [1], utilizando el lenguaje Hugs, y de ella se concluyó que un lenguaje de programación de propósito general, presenta las siguientes desventajas:

- la dificultad de aprender dos lenguajes (el matemático y el de programación) en el mismo curso
- los estudiantes deben "manipular" los objetos matemáticos como tales para su comprensión y no como son presentados en un lenguaje de programación, ya que la aplicación del mismo, no está orientada a la enseñanza.

De esta forma, la introducción de un lenguaje de programación en la cual se refleje la teoría que se está presentando en forma "pura", parece ser el mejor camino para lograr una real integración entre matemática y computación. Dos de estas herramientas son: el lenguaje de programación matemático ISETL (http://csis03.muc.edu/isetlw/isetlw.html), y MAC (http://universitario.edu.uy/MAC) actualmente en desarrollo a cargo de los autores de este artículo.

Aún cuando pensamos que estos esfuerzos son positivos y útiles, consideramos que no son suficientes, son intentos aislados de resolver un problema que es común tanto para la enseñanza de matemática como de computación. Ni el uso de software o lenguajes de programación en cursos de matemática como meras herramientas ni la desconsideración de las deficiencias en la base matemática de los estudiantes de ciencia de la computación, ayudará de forma efectiva en la solución del problema. Por el contrario, lo que se necesita es, en nuestra opinión, el trabajo conjunto de docentes de ambas disciplinas para crear y enseñar nuevos cursos a ser introducidos como parte de la formación básica requerida para seguir estudios científicos o tecnológicos. En dichos cursos, la matemática discreta debe jugar un papel central y lo que es aún más importante, la relación entre ella y ciencia de la computación debe ser enfatizada explícitamente.

Casi todo el tiempo los cursos de matemática de Secundaria se dedican al estudio de Cálculo. De esta manera, los estudiantes creen que matemática es la matemática continua y que todo se relaciona con el campo de los números reales. Hemos entrevistado grupos de estudiantes del primer año de ciencia de la computación en la universidad, encontrando resultados que confirman este hecho, por ejemplo, nunca describen una función incluyendo dominio y codominio (todo es R), casi nunca usan correctamente los cuantificadores en las pruebas o nociones de lógica en los cálculos (lo que cuenta es el cálculo) y tratan el método de inducción como una receta para resolver algunos problemas, generalmente sobre sumatorias de números.

En nuestro enfoque, demostramos cuánto puede hacerse sin el campo de los números reales, para contrarrestar la idea de que la matemática es siempre continua.

2 El enfoque

Los principales objetivos de nuestra actividad son:

- Explicar la relación entre ciencia de la computación y matemática discreta a través de conceptos comunes a ambas disciplinas, como funciones, algoritmos, definiciones inductivas, etc.
- Mostrar los beneficios de incluir un lenguaje de programación: se requiere un tratamiento riguroso de las definiciones y permite "olvidar" los cálculos, ayudando a desarrollar otras habilidades como el proceso de resolver un problema pensando en qué debe hacerse y no cómo debe hacerse.
- Enfatizar el proceso de abstracción a través de la construcción paso a paso de la solución de un problema a partir de su especificación
- Remarcar la importancia de la lógica para comprender los conceptos de definición, prueba, propiedad, etc.
- Demostrar que la computación involucra actividades matemáticas interesantes.

Dos tópicos de nuestro curso merecen especial atención y discusión: el concepto de función y el concepto de inducción.

A continuación resumimos los principales argumentos debatidos, porque creemos que aclaran la diferencia entre nuestro enfoque y el tradicional.

Los docentes de matemática reconocen ampliamente el hecho de que la mayoría de los estudiantes tienen el concepto de función restringido a la idea de una fórmula algebraica para computar valores[2]. Sin esta fórmula son verdaderamente reacios a reconocer que alguna función pueda estar presente. De ahí las dificultades en tratar a las funciones como objetos que pueden ser manipulados, operados, aplicados a otras funciones, etc. Esta noción incompleta de función tiene efectos negativos para aprender conceptos de ciencia de la computación y de matemática, y desafortunadamente, esta fuertemente arraigada en la mente de los estudiantes. Sin embargo, una función es usualmente introducida como una relación especial, es decir como un subconjunto de un producto cartesiano de dos conjuntos. Asociada a esta definición, suele hacerse la representación gráfica usando flechas entre los elementos de los conjuntos. Esta noción introductoria de función está muy cercana al concepto de función como proceso, las flechas representan una transformación, una regla de correspondencia, el dominio y el recorrido están presentes, más aún, no tienen por qué ser conjuntos numéricos. Ninguna fórmula algebraica ha aparecido. De esta manera, los conceptos de composición de funciones, funciones de alto orden, algoritmos, etc, deberían derivarse naturalmente.

Pero, este camino natural de introducir el concepto de función, rápidamente se desvía hacia un **caso particular que pasa a dominarlo todo:** las funciones cuyo dominio y codominio son subconjuntos de los números reales y existe una fórmula algebraica para computar valores.

Este caso particular pertenece a los cursos de cálculo y debido al excesivo peso de esta rama de la matemática en la enseñanza, los estudiantes "olvidan" el concepto original de función.

El otro tópico discutido es métodos de prueba en general y pruebas por inducción en particular y su relación con la lógica, lo cual es raramente considerado en Enseñanza Secundaria.

Al hacer pruebas por inducción los estudiantes piensan en una "receta" para probar determinadas propiedades, generalmente con sumatorias.

En nuestro curso presentamos el concepto de inducción relacionado con [1]:

- La definición inductiva de conjuntos (conjuntos inductivos).
- La definición de funciones sobre dominios inductivos, es decir, usando patterns y ecuaciones.
- La prueba de propiedades sobre conjuntos inductivos, usando el principio de inducción estructural correspondiente.

Proponemos ejercicios sobre diferentes conjuntos inductivos, por ejemplo, las fórmulas bien formadas del cálculo proposicional, las secuencias de caracteres de algún alfabeto, etc.

3 Marco teórico

La metodología que adoptamos es similar a [3], en el sentido de que los estudiantes trabajan en actividades antes de la presentación formal de los conceptos. Esta metodología corresponde a un marco teórico que nosotros compartimos y que se basa en la teoría Epistemología Genética de Jean Piaget, sobre la construcción del conocimiento por los individuos [4]. Su centro de interés es la descripción del desarrollo de los esquemas cognitivos de los individuos a lo largo del tiempo y de acuerdo con ciertas reglas generales. La construcción de dichos esquemas se basa en la interacción del sujeto y el medio, que da lugar a un proceso de dos sentidos:

- la *asimilación*, por la cual el sujeto intenta asimilar una situación nueva a sus estructuras cognictivas existentes y
- la *acomodación* por la cual esas estructuras se expanden y se reconstruyen para acomodarse a la nueva situación.

De esta forma se logra un ***equilibrio*** cognicitivo que es dinámico, en el sentido de que el proceso continúa por interacción con nuevas situaciones posibles. La fuente de la interacción y por lo tanto de todo el proceso cognictivo es la acción del sujeto y el mecanismo por el cual las estructuras cognictivas se reconstruyen ha sido definido por Piaget como la ***abstracción reflexiva.*** Por medio de este mecanismo, lo realizado en el plano de la acción es reflejado al plano superior de la conceptualización, en el cual se produce una coordinación e interiorización de las acciones, a la cual Piaget llama operaciones, dando lugar a una estructura cognictiva más general [5]. La importancia del papel jugado por la abstracción reflexiva en la construcción de los conceptos matemáticos ha dado lugar, recientemente, a dos marcos teóricos, extensiones de la teoría desarrollada por Jean Piaget: La generalización operativa [6] y el marco teórico acción-proceso-objeto [7] del cual adoptamos nuestra metodología.

Sin embargo, nuestra aplicación de la metodología difiere con [3] en el sentido de que nosotros siempre requerimos que los estudiantes realicen actividades matemáticas exploratorias de conceptos y propiedades, que luego se formalizan y programan en ISETL, mientras que en [3] las actividades exploratorias de realizan usando ISETL

Creemos que de esa manera, la relación entre matemática y ciencia de la computación se establece más claramente, al incorporar la fase de programación en el proceso de resolución de un problema.

4 Breve descripción de MAC

MAC es un lenguaje interpretado, que surge como una solución a las conclusiones expuestas en [8], y como herramienta de apoyo didáctico para la integración de matemática y programación en cursos de matemática discreta. Esta herramienta permite la definición y manipulación de objetos matemáticos finitos en un lenguaje similar al matemático, y la visualización gráfica de estos objetos cuando corresponda. Presentaremos a continuación la definición, evaluación y manipulación de funciones en MAC, concepto central en la teoría de la computación.

4.1 Definición de funciones

En MAC es posible definir funciones sobre conjuntos finitos, utilizando un lenguaje similar al empleado en el curso de Matemática Discreta. Cuando se define la función se debe explicitar el dominio y codominio, y se puede realizar:

- explicitando el conjunto de pares ordenados
- por casos
- al estilo cálculo lambda

Ejemplos de definiciones de funciones en MAC:

```
deff  f:{1,2,3}->{"a","b","c"};
f={<1,"a">,<2,"b">,<3,"a">}
-- definición como un conjunto de pares

deff g: {1,2,3}->{"a","b","c"};
g x = case x of
    1 -> "c",
    2 -> "b",
    3 -> "a"
fo
-- definición por casos
```

deff h:{1,2}->{2,3,4};
h=[x] x + 1
-- definición al estilo del cálculo lambda

4.2 Evaluación de expresiones

La aplicación de la función se realiza mediante la evaluación de expresiones, independientemente de como dicha función haya sido definida. Esto es particularmente importante. En ISETL, si bien puede definirse una función como un conjunto de pares, luego es tratado como un conjunto y no como una función, por lo tanto no es posible realizar una aplicación.

También pueden evaluarse en MAC propiedades de la función como inyectividad, sobreyectividad, etc., mientras que en ISETL esto no es posible, dado que no se explicita el dominio y codominio de las funciones.

Ejemplos de evaluación de expresiones en MAC:

eval f(1) -- el resultado de aplicar f al valor 1
resultado: "a"

eval h(1)
resultado: "a"

eval g(1)
resultado: 2

eval esiny(f) --evalúa si f es una función inyectiva
resultado: False

eval essob(g) -- evalúa si g es una función sobreyectiva
resultado: True

eval ((inv(g)) o f) (1) -- evalúa la composición de f con la inversa de g,
-- aplicada a 1
resultado: 3

eval (inv(g)) -- evalúa la aplicación de inv a la función g
resultado: {[{"a","b","c"},{1,2,3}],{("c",1),("b",2),("a",3)}}
-- se explicita el dominio, codominio y el conjunto de pares.

4.3 Representación gráfica

En el caso de MAC, el estudiante tiene la posibilidad de visualizar gráficamente el resultado de la evaluación de expresiones o de la definición de funciones, conjuntos, etc.

Ejemplos de representaciones gráficas:

Los objetos que se definan o sean resultados de expresiones, pueden ser representados gráficamente en MAC, como se muestra en la Fig. 1.

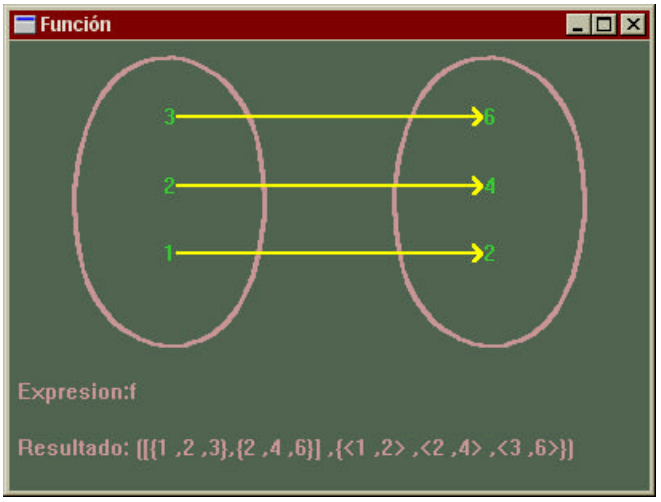

Fig. 1 Representación gráfica de una función

Presentamos a continuación un ejemp lo sobre una de las mayores desventajas que consideramos que presenta ISETL:

Sea la función f: {1,2,3}-> {1,2,3,4,5,6}tal que f(x)=2*x.

En Isetl, esta función puede definirse de la siguiente manera:

Definición 1:

```
f := func(x);
        return 2*x
end;
```

Pero esta representación de f en ISETL, es en realidad, una familia de funciones, que incluye todas aquellas funciones que están definidas sobre todos los conjuntos para los cuales la operación (2*) está definida, y no para el dominio especificado. Por tal motivo, no pueden testearse operaciones como: determinar si f es inyectiva, determinar si f es total. Sin embargo, si se representa gráficamente esta función en IsetL, se asume que el dominio y codominio es el conjunto de los números reales, como se muestra en la Fig. 2.

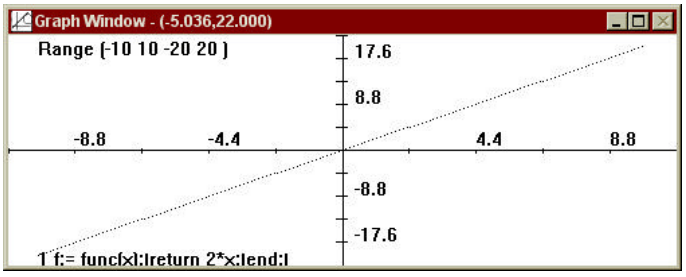

Fig. 2 Representación gráfica de una función en IsetL

Esto constituye también un claro ejemplo de lo que afirmamos en la Introducción sobre el peso excesivo del Cálculo y sus efectos negativos sobre la comprensión del concepto de función: en realidad no se considera importante incluir el dominio y el codominio porque se asume que es R.

En Mac, esta función puede definirse como:

Definición 1:

deff f:{1,2,3}->{1,2,3,4,5,6}; f={<1,2>,<2,4>,<3,6>}

En la definición se explicita que es una función (deff). Por tal motivo, pueden aplicarse a f las operaciones que pueden aplicarse a funciones: determinar si es inyectiva (eval esiny(f)), si es sobreyectiva (eval essob(f)) y la aplicación de f a algún elemento del dominio (eval f(1)).

Definición 2:

deff f:{1,2,3}->{1,2,3,4,5,6};

f=[x] x * 2

En este caso, la definición de la función se realiza como una abstracción, y pueden aplicarse las mismas operaciones que en el caso anterior.

5 Conclusiones y Trabajos Futuros

Nuestro curso requirió como trabajo final de los participantes (profesores de matemática de la enseñanza media), realizar una experiencia consistente en introducir en clases liceales con alumnos de 16 años, temas de Matemática Discreta, como Divisibilidad, Combinatoria, Pruebas por inducción, etc., enfocados de acuerdo a lo aprendido en nuestro curso y utilizando el lenguaje ISETL. Todos los docentes coincidieron en que la actitud de los estudiantes hacia el uso del lenguaje fue muy positiva y que no ofreció ninguna dificultad. También hubo acuerdo en el hecho de que al resolver los problemas partiendo de una especificación de alguna solución y derivar paso a paso la formulación matemática y luego el programa en ISETL, los estudiantes lograban aplicar un proceso de abstracción que les permitía fácilmente llegar a las definiciones de funciones.

Nosotros confiamos que el lenguaje Mac mantendrá estas características de los resultados evaluados y mejorará otros, por ejemplo, en lo que se refiere a las definiciones de funciones y en cuanto a las representaciones gráficas.

Las primeras experiencias con el uso de este software en un curso de matematica se realizarán al finalizar la próxima fase de desarrollo, cuyo objetivo es extender las posibilidades de definir conjuntos finitos (que actualmente es por extensión), con la incorporación de definiciones de conjuntos finitos por comprensión, incorporando además, el concepto de predicado .Una de estas experiencias se realizará con docentes de Enseñanza Secundaria, en forma similar a la realizada con ISETL, y la otra será llevada a cabo con un grupo de estudiantes de los cursos de matemática discreta del primer año de una carrera informática.

6 Referencias

[1] Sylvia da Rosa y Gustavo Cirigliano, *Matemática y Programación,* anales del V Congreso Iberoamericano de Educación Superior en Computación, Ciesc 1997.

[2] Dubinsky et al., *A Framework for Research and Curriculum Development in Undergraduate Mathematics Education*, work partially supported by the National Science Foundation, Division of Undergraduate Education, 2000.

[3] E.Dubinsky and W. Fenton, *Introduction to Discrete Mathematics with ISEL*, Springer Verlag, 1996.

[4] J. Piaget y R. García, *Psicogénesis e Historia de las Ciencias*, 1982.

[5] J.Piaget y E.Beth, *Epistemología Matemática y Psicología*, 1969.

[6] W.Dörfler, *Forms and Means of Generalization in Mathematics*, in Mathematical Knowledge: Its growth through teaching, 1991.

[7] E.Dubinsky, P.Lewin, *Reflective Abstraction and mathematics Education: The Genetic Decomposition of Induction and Compactness*, The Journal of Mathematics Behaviour, 1986.

[8] Sylvia da Rosa y Gustavo Cirigliano, *Ensayo sobre Matemática aplicada a Computación*, anales del VI Congreso Iberoamericano de Educación Superior en Computación, Ciesc 1998.

[9] IEEE Computer Society. Asociation for Computing Machinery. *Computing Curricula 2001. Computer Science.* 2001.

[10] Viera K. Proulx. The *Role of Computer Science and Discrete Mathematics in the High School Curriculum.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science , Vol 36 (1997).